```
1: # VexCode-2020
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jun  7 10:22:26 2019

@author: aiden
"""

import os
import treelib


class DirectoryTree:
    """
    contains methods for making a treelib tree structure of a
    given directory tree
    meant to be inherited but can be a stand alone class

    @params = None
    @return = None
    """

    def __init__(self):
        self.__directories = [] #will contain all directories and subdirectories
        self.__directory_tree = treelib.Tree() #will contain all directories ordered according
                                #to "config.txt"



    def __get_dirs(self, parent_directory, storage_list):
        """
        recursively prints all files in parent directory
        put in list given

        @params = directory to start at, directory to write output to
        @return = None
        """

        dirs = [f.path for f in os.scandir(parent_directory) if f.is_dir()]

        for directory in dirs:
            storage_list.append(directory)
            self.__get_dirs(directory, storage_list)



    def __get_all(self, parent_directory, storage_list):
        """
        recursively prints all files in parent directory
        put in list given

        @params = directory to start at, directory to write output to
        @return = None
        """

        dirs = [f.path for f in os.scandir(parent_directory) if f.is_dir()]
        files = [f.path for f in os.scandir(parent_directory)]

        for file in files:
            storage_list.append(file)

        for directory in dirs:
            storage_list.append(directory)
            self.__get_all(directory, storage_list)



    def __make_tree(self):
        """
        add directories found to a treelib structure
        the parent directory is set in "config.txt"

        @params = None
        @return = None
        """
        depth = self.__directory_tree.depth()

        #splits path of directory and makes nodes of partial paths until
        #it reaches the end of the path given
        #the resulting tree is saved in "self.__directory_tree"
        for path in self.__directories:
            path_split = path.split("/")

            i = depth

            while i < len(path_split):
                try:
                    if i > 0:
                        name = path_split[i]
                        node_id = "/".join(path_split[0:i + 1])
                        parent = "/".join(path_split[0:i])
                    else:
                        name = path_split[i]
                        node_id = path_split[i]
                        parent = None

                    self.__directory_tree.create_node(tag=name, identifier=node_id, parent=parent)

                except treelib.exceptions.NodeIDAbsentError:
                    pass
                except treelib.exceptions.DuplicatedNodeIdError:
                    pass

                i = i + 1



    def return_directory_tree(self, parent_directory):
        """
        makes and returns a treelib tree structure of a directory
        tree based on a parent directory given only directories are
        included
```

```
114:
115:        @params = string of parent directory to start node at
116:        @return = treelib tree structure of directory tree,
117:                  list of directories found
118:        """
119:        self.__directory_tree = treelib.Tree()
120:        self.__directories = []
121:
122:        self.__get_dirs(parent_directory, self.__directories)
123:        self.__make_tree()
124:
125:        return self.__directory_tree, self.__directories
126:
127:
128:
129:    def return_tree(self, parent_directory):
130:        """
131:        makes and returns a treelib tree structure of a directory
132:        tree based on a parent directory given
133:        everything is included
134:
135:        @params = string of parent directory to start node at
136:        @return = treelib tree structure of directory tree,
137:                  list of directories found
138:        """
139:        self.__directory_tree = treelib.Tree()
140:        self.__directories = []
141:
142:        self.__get_all(parent_directory, self.__directories)
143:        self.__make_tree()
144:
145:        return self.__directory_tree, self.__directories
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Jun  5 19:13:03 2019

@author: aiden
"""

import math
import os
import shutil


class PostScript:
    """
    class for converting code to pictures
    """
    def __init__(self):
        self.alldirs = [".."] #list declaration that will contain all
                              #directories. Parent directory is included in it
                              #because there is code to be printed in it
                              #
                              #list will contain all directories that contain
                              #files to be printed

        self.allfiles = [] #list declaration that will contain all files that
                           #must be converted to postscript

        self.pictures_dir = "../Pictures/" #parent directory of where pictures
                                           #will be placed

        self.pictures = [] #contains locations of pictures

        self.directory_exceptions = [ #directories that contain no files to
                                      #be printed
            "../AutonSimulator/__pycache__",
            "../Coach/__pycache__",
            "../CodeGenerator/__pycache__",
            "../HaarCascadeTraining",
            "../Pictures",
            "../PIDDebugging/__pycache__",
            "../Print/__pycache__",
            "../Prototypes/queue/unit_test/log",
            "../Prototypes",
            "../Prototypes/queue/unit_test/date",
            "../Prototypes/queue/unit_test/analysis",
            "../Prototypes/EmbeddedQueue/unit_test/log",
            "../Prototypes/EmbeddedQueue/unit_test/analysis",
            "../Prototypes/EmbeddedQueue/unit_test/date",
            "../Prototypes/SegfaultTest/date",
            "../Prototypes/SegfaultTest2/date",
            "../RobotCode/bin",
            "../RobotCode/firmware",
            "../RobotCode/include/display",
            "../RobotCode/include/okapi",
            "../RobotCode/include/pros",
            "../RobotCode/lib",
            "../RobotCode/.d",
            "../RobotCode/src/JSONLibrary",
            "../RobotCode/src/objects/lcdCode/fonts",
            "../Scouting/__pycache__",
            "../Cascades",
            "../.git"
            ]

        self.file_exceptions = [  #files in included directories of acceptable
                                  #extensions that should not be converted to
                                  #postscript to be printed
            "../license.html",
            "../PIDDebugging/ut.txt",
            "../PIDDebugging/test.txt",
            "../Print/tree.txt",
            "../Print/tree.rtf",
            "../Prototypes/queue/concurrentqueue.h",
            "../Prototypes/queue/LICENSE.md",
            "../Prototypes/queue/blockingconcurrentqueue.h",
            "../Prototypes/JSONReader/json.hpp",
            "../RobotCode/include/main.h",
            "../RobotCode/include/api.h",
            "../RobotCode/compile_commands.json"
            ]

        self.valid_extensions = [ #files with this extension will be allowed to
                                  #be printed
            ".py",
            ".c",
            ".cpp",
            ".hpp",
            ".h",
            ".sh",
            ".txt",
            ".json",
            ".md"
            ]

        if not os.path.isdir(self.pictures_dir):
            os.mkdir(self.pictures_dir)  #makes directory for pictures


    def __get_directories_recursively(self, parent_directory, storage_list):
        """
        recursively prints all files in parent directory
        put in list "alldirs"

        @params = directory to start at, directory to write output to
        @return = None
        """

        dirs = [f.path for f in os.scandir(parent_directory) if f.is_dir()]

        for directory in dirs:
            storage_list.append(directory)
            self.__get_directories_recursively(directory, storage_list)
```

```
114:
115:
116:
117:    def __find_files(self, directory):
118:        """
119:        finds all files that are not subdirectories in a given directory that
120:        are of a specific type
121:
122:        @params = directory to look through
123:        @return = type list of files that are in the directory that are
124:                not sub directories
125:        """
126:
127:        files = [] #contains all files that will be returned
128:
129:        contents = [f.path for f in os.scandir(directory) if not f.is_dir()]
130:            #returns everything in directory including
131:            #sub directories
132:
133:        for file in contents: #checks to make sure that only files that are not
134:                              #directories and have a valid extension are added
135:                              #to the list of files
136:            _, extension = os.path.splitext(file)
137:            if extension in self.valid_extensions:
138:                files.append(file)
139:
140:        return files
141:
142:
143:
144:
145:
146:    def __convert_to_postscript(self, file):
147:        """
148:        converts the specified file to postscript format which is a printable
149:        code type. The files are placed inside of "../pictures/" in the same
150:        directories that they were originally in
151:
152:        @params = relative path of file to convert
153:        @return = None
154:        """
155:
156:        _, extension = os.path.splitext(file)
157:        output_name = ((self.pictures_dir + file.split(extension)[0]
158:                        + extension.upper().split(".")[1]).replace("/../", "/")
159:                       + ".ps")
160:
161:        self.pictures.append(output_name)
162:
163:        #get the highlight color that will be used by shell command
164:        #based on the type of file it is
165:        if extension == ".py":
166:            highlight_color = "--highlight=python"
167:
168:        elif extension in [".cpp", ".hpp"]:
169:            highlight_color = "--highlight=cpp"
170:
171:        elif extension in [".c", ".h"]:
172:            highlight_color = "--highlight=c"
173:
174:        elif extension == ".sh":
175:            highlight_color = "--highlight=bash"
176:
177:        else:
178:            highlight_color = "--highlight=mail"
179:
180:        #runs command that will convert a file to postscript
181:        #the output file location is in self.pictures
182:        os.system("enscript -G --line-numbers -o "
183:                  + output_name + " "
184:                  + highlight_color
185:                  + " --color=1 -f Palatino-Roman5 --columns 1 "
186:                  + file)
187:
188:
189:    def prepare_code(self):
190:        """
191:        prepares code to be printed by converting all code to
192:        postscript
193:
194:        @params = None
195:        @return = None
196:        """
197:        self.allfiles = [] #list declaration that will contain all files that
198:                           #must be converted to postscript
199:
200:        self.pictures_dir = "../Pictures/" #parent directory of where pictures
201:                                           #will be placed
202:
203:        self.pictures = [] #contains locations of pictures
204:        shutil.rmtree(self.pictures_dir)  #clean out pictures folder so that
205:                                          #nothing is there that shouldn't be
206:        os.mkdir(self.pictures_dir)
207:
208:        self.__get_directories_recursively("..", self.alldirs)
209:                           #gets unrefined list of directories
210:                           #saved in list "alldirs"
211:        to_remove = []
212:        for directory in self.alldirs: #iterates through each directory
213:                                       #found
214:            for exception in self.directory_exceptions: #checks to see if
215:                                                        #directory should
216:                                                        #be excluded
217:                if exception in directory:
218:                    to_remove.append(directory)  #do not remove items from list
219:                                                 #because changing the size
220:                                                 #while iterating through the
221:                                                 #list will cause
222:                                                 #unexpected results
223:
224:                    break #end looking for exception because if one has
225:                          #been found it is already in the to remove list
226:                          #and no other cases will match it
```

```
227:
228:        alldirs = set(self.alldirs) #convert each list to type set
229:                                    #and find the difference between them
230:                                    #the difference will always be the
231:                                    #directories wanted because all the items
232:                                    #in "to_remove" will be in "alldirs" and all
233:                                    #that is left is the directories to keep
234:
235:        to_remove = set(to_remove)
236:
237:        alldirs = list(alldirs - to_remove) #find difference of the two
238:                                            #sets and convert back to type
239:                                            #list remaining list will be
240:                                            #the directories that contain
241:                                            #code to be printed
242:
243:
244:
245:
246:
247:        for directory in alldirs:
248:            self.allfiles.append(self.__find_files(directory))
249:                                            #by adding
250:                                            #the list
251:                                            #directly to the all
252:                                            #files list, list will
253:                                            #be segmented and can
254:                                            #be printed in order
255:                                            #easier
256:
257:            #make directory under pictures directory for a picture of the code
258:            #to go
259:            if not os.path.isdir((self.pictures_dir + directory).replace("/../", "/")):
260:                os.makedirs((self.pictures_dir + directory).replace("/../", "/"))
261:
262:        for fileset in self.allfiles:
263:            for file in fileset:
264:                if not file in self.file_exceptions:
265:                    self.__convert_to_postscript(file)
266:
267:
268:
269:    def get_page_count(self, files=None):
270:        """
271:        gets total size of all post script files in
272:        "Pictures" directory that was made on config file
273:        in "Pictures" directory
274:
275:        @params = (optional) type list of paths of files to print
276:        @return = int amount of pages
277:        """
278:        if files:
279:            self.pictures = files
280:
281:        elif not self.pictures and not files:
282:                            #if pictures list contains any file locations
283:                            #and if no files have been specified
284:                            #if it does not, then program will find all the
285:                            #files that are currently in the directory
286:                            #useful if user does not want to overwrite
287:                            #what is currently there
288:
289:            self.pictures.append(self.pictures_dir) #add parent directory
290:                                                    #because it will not be
291:                                                    #there to start
292:            self.__get_directories_recursively(self.pictures_dir, self.pictures)
293:
294:            self.valid_extensions.append(".ps") #temporarily add postscript to
295:                                                #acceptable extensions so that the
296:                                                #function used will not exclude it
297:                                                #entry is removed later
298:            picture_files = []
299:            for directory in self.pictures: #find all files that are not
300:                                            #subdirectories
301:                for file in self.__find_files(directory):
302:                    picture_files.append(file)
303:
304:            self.valid_extensions.remove(".ps") #remove postscript from
305:                                                #acceptable extensions because
306:                                                #it is no longer needed
307:
308:            self.pictures = picture_files #"self.pictures" now contains
309:                                          #locations of files
310:
311:        total_pages = 0
312:        for picture in self.pictures: #iterate through and parse each file
313:                                      #looking for a keyword
314:                                      #keyword appears twice once with
315:                                      #parenthases and once without, so
316:                                      #algorithm makes sure it contains no
317:                                      #parenthases
318:            file = open(picture, "r")
319:            for line in file.readlines():
320:                if "%%Pages: " in line and "(" not in line:
321:                    total_pages = (total_pages + math.ceil(int(line.split("%%Pages: ")[1].split("\n")[0]) / 2))
322:
323:        return total_pages
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Jun  6 15:20:10 2019

@author: aiden
"""

import os
import treelib

import DirectoryTree


class Print(DirectoryTree.DirectoryTree): #inherits DirectoryTree so that
                                #a directory tree can be made
    """
    contains methods for printing code based on
    "config.txt"
    """
    def __init__(self):
        DirectoryTree.DirectoryTree.__init__(self)

        self.directory_tree = treelib.Tree() #will contain all directories ordered according
                                    #to "config.txt"

        self.swap_tree = treelib.Tree() #used to re-arrange directories in
                                #other node

        self.parent = "." #will contain parent to append to file paths
                        #in "config.txt"
        self.headers_only = 0 #option set in "config.txt"
        self.headers_first = 1 #option set in "config.txt"

        self.dir_order = [] #will contain order of directories set in "config.txt"
        self.file_order = [] #will contain order of files set in "config.txt"

        self.print_order = [] #will contain paths of all print items in order



    @classmethod
    def __get_file_type(cls, path):
        """
        takes a file converted to postscript and finds what file type it
        is (.cpp, .hpp, .sh)

        @params = path name of file to get the type of
        @return = type string of extension (ex. ".cpp")
                type string of root path (ex. no directories and file type (CPP) cut off)
                type string of path without root path
        """
        valid_extensions = [ #files with this extension will be allowed to
                        #be printed
            "PY",
            "C",
            "CPP",
            "HPP",
            "H",
            "SH",
            "TXT",
            "JSON",
            "MD"
            ]

        filename, _ = os.path.splitext(path) #remove extension
        filename = filename.split("/") #remove other directories
        filename = filename[len(filename) - 1]

        one_char = filename[-1:]
        two_char = filename[-2:]
        three_char = filename[-3:]
        four_char = filename[-4:]

        root_name = ""

        if one_char in valid_extensions and two_char not in valid_extensions:
            #added checking for two char as well to fix cases SH and H
            extension = one_char
        elif two_char in valid_extensions:
            extension = two_char
        elif three_char in valid_extensions:
            extension = three_char
        elif four_char in valid_extensions:
            extension = four_char
        else:
            extension = "invalid"

        if extension != "invalid":
            root_name = filename.split(extension)[0]

        root_path = path.split(root_name + extension)[0]

        return extension, root_name, root_path



    def __get_rules(self):
        """
        gets rules based on "config.txt"

        @params = None
        @return = None
        """
        #extract parent from "config.txt"
        with open("config.txt", "r") as config: #file closed at end of block
            for line in config.readlines():
                #makes sure all required elements are in the line
                #and that it is not a comment
                if "parent" in line and "=" in line and "#" not in line:
                    self.parent = line.split("parent")[1].strip().strip('\"')
```

```
114:                self.parent = self.parent.split("=")[1].strip().strip('\"')
115:
116:        #extract other parameters from "config.txt"
117:        param_names = ["HEADERS_FIRST", "ONLY_HEADERS"]
118:        with open("config.txt", "r") as config: #file closed at end of block
119:            for line in config.readlines():
120:                #makes sure all required elements are in the line
121:                #and that it is not a comment
122:                if param_names[0] in line and "#" not in line:
123:                    line = line.split(param_names[0])[1].strip()
124:                    self.headers_first = int(line)
125:
126:                elif param_names[1] in line and "#" not in line:
127:                    line = line.split(param_names[1])[1].strip()
128:                    self.headers_only = int(line.strip())
129:
130:        #read rules
131:        with open("config.txt", "r") as config: #file closed at end of block
132:            for line in config.readlines():
133:                if '#' not in line:
134:                    if "dir" in line:
135:                        #convert to standard os path
136:                        directory = self.parent + "/" + (line.split("dir ")[1].rstrip())
137:                        directory = os.path.normpath(directory)
138:                        self.dir_order.append(directory)
139:                    elif "file" in line:
140:                        #convert to standard os path
141:                        file = self.parent + "/" + (line.split("file ")[1].rstrip())
142:                        file = os.path.normpath(file)
143:                        self.file_order.append(file)
144:
145:
146:
147:
148:    def __order_directories(self):
149:        """
150:        orders directories in "self.directory_tree"
151:        based on settigns in "config.txt"
152:        makes new tree stored in "self.swap_tree"
153:
154:        @params = None
155:        @return = None
156:        """
157:        #iterate through each branch of the tree and sort it
158:        #moving directories to needed spot in "self.swap_tree"
159:        #standard path will be made from os module
160:        #so that no conflicts occur from directory separators
161:        #a two number string is added to all the tags in the node so that
162:        #the order is maintained. The order for nodes is in
163:        #alphabetical order, so no changes would occur if the numbers were
164:        #not added
165:
166:        #finds nodes in tree that have children
167:        parent_nodes = []
168:        for node in self.directory_tree.all_nodes():
169:            if not node.is_leaf(): #a leaf has no children
170:                parent_nodes.append(node)
171:
172:        #adds the root node to the swap tree
173:        self.swap_tree.create_node("00" + parent_nodes[0].tag,
174:                                   parent_nodes[0].identifier)
175:        parent_nodes.pop(0)
176:
177:        #sorts children of parent node
178:        #
179:        #a string of two numbers is added to the tag so that the tree is
180:        #in order, the id is not affected however
181:        #this means the max amount of directories in one node can only be 99
182:        #this should probably not be exceded
183:        for node in parent_nodes:
184:            num = 0
185:            children = self.directory_tree.children(node.identifier)
186:            try: #adds parent of the node to the tree if it is not already there
187:                self.swap_tree.create_node("00" + node.tag,
188:                                           node.identifier,
189:                                           self.directory_tree.parent(node.identifier).identifier)
190:            except treelib.exceptions.DuplicatedNodeIdError: #node already exists
191:                pass
192:
193:            #get applicable rules
194:            applicable_rules = [] #contains identifier for nodes in order to
195:                                  #be sorted
196:            for rule in self.dir_order:
197:                dir_length = len(rule.split("/")) #checks to see if amount of
198:                                                  #directories in path is the
199:                                                  #same as the amount in the
200:                                                  #tree
201:                                                  #if it is, it is added to
202:                                                  #applicable rules
203:
204:                offset = len(self.parent.split("/")) #offset is added to the
205:                                                     #tree level because the
206:                                                     #rules in "config.txt" do
207:                                                     #not contain the parent
208:                                                     #directory
209:                children_names = list(map(lambda x: x.identifier, children))
210:                tree_level = self.directory_tree.level(node.identifier) + offset
211:                if rule in children_names and dir_length == tree_level:
212:                    applicable_rules.append(rule)
213:
214:            for child in applicable_rules: #add nodes to the tree that appear
215:                                           #in the given rules so that the order
216:                                           #wanted is kept
217:                num = str(num) #makes sure that "num" is two characters
218:                if len(num) < 2:
219:                    num = "0" + str(num)
220:
221:                nid = self.directory_tree.get_node(child)
222:                self.swap_tree.create_node(str(num) + nid.tag,
223:                                           nid.identifier,
224:                                           node.identifier)
225:                num = int(num) + 1
226:                children.remove(self.directory_tree.get_node(child))
```

```
227:
228:            while children: #adds other children to the new tree
229:                    #order does not matter so they are added in order
230:                    #of appearance
231:                num = str(num) #makes sure that "num" is two characters
232:                if len(num) < 2:
233:                    num = "0" + num
234:
235:                self.swap_tree.create_node(str(num) + children[0].tag,
236:                                           children[0].identifier,
237:                                           parent=node.identifier)
238:                num = int(num) + 1
239:                children.pop(0)
240:
241:
242:
243:
244:        def __order_headers(self, files):
245:            """
246:            takes a list of postscript files and orders it so that headers
247:            appear before implementation files in the list
248:
249:            @params = list of postscript files to sort putting headers before
250:                    implementation file
251:            @return = sorted list of postscript files
252:            """
253:            #create dictionary with key of the root file name and then a value of
254:            #a list of the extensions with that root name
255:            file_dict = {}
256:
257:            for file in files:
258:                extension, root_name, root_path = self.__get_file_type(file)
259:                if root_name not in file_dict.keys():
260:                    file_dict.update({root_name : [extension]})
261:                else: #if key exists then add value to the list
262:                    file_dict[root_name].append(extension)
263:
264:            #iterate through dictionary placing hpp files ahead of cpp
265:            for extension_list in file_dict.values():
266:                extension_list.sort(reverse=self.headers_first)
267:
268:            #re construct list
269:            ordered_files = []
270:            for item in file_dict:
271:                extensions = file_dict.get(item)
272:                for extension in extensions:
273:                    if extension != "invalid":
274:                        if not self.headers_only:
275:                            file = root_path + item + extension + ".ps"
276:                            ordered_files.append(file)
277:                        else: #if only headers are to be printed
278:                            if extension != "CPP" or len(extensions) == 1:
279:                                file = root_path + item + extension + ".ps"
280:                                ordered_files.append(file)
281:
282:            return ordered_files
283:
284:
285:
286:
287:        def __order_files(self):
288:            """
289:            creates a list of file names in the order that they are to be printed
290:
291:            @params = None
292:            @return = None
293:            """
294:            #get directories to look through
295:            directories = list(self.swap_tree.expand_tree())
296:            ordered_files = [] #holds list of all ordered files
297:
298:            for directory in directories:
299:                ordered_directory = [] #holds list of ordered files in a directory
300:
301:                #gets the files in a given directory
302:                files = list([f.path for f in os.scandir(directory) if not f.is_dir()])
303:                files.sort() #makes paths in aphabetical order
304:
305:                #gets rules for files in a directory
306:                applicable_rules = []
307:
308:                for rule in self.file_order:
309:                    rule_directory = rule
310:                    rule_directory = rule_directory.split("/")
311:                    rule_directory.pop(-1)
312:                    rule_directory = "/".join(rule_directory)
313:
314:                    rule_dirs = len(rule.split("/")) - 1
315:
316:                    #add one to the tree level because the level starts at 0 instead
317:                    #of 1 like the method to find the height of other directories
318:                    tree_level = self.swap_tree.level(self.swap_tree.get_node(directory).identifier) + 1
319:
320:                    #if rule applies
321:                    if rule_directory == directory and rule_dirs == tree_level:
322:                        applicable_rules.append(rule)
323:
324:                #adds files where a specified order has been given
325:                for file in applicable_rules:
326:                    ordered_directory.append(file)
327:                    files.remove(file)
328:
329:                #adds rest of files
330:                while files:
331:                    ordered_directory.append(files[0])
332:                    files.pop(0)
333:
334:                #sorts so that headers are first if that option
335:                #has been given
336:                #also checks if only headers will be printed
337:                ordered_directory = self.__order_headers(ordered_directory)
338:
339:                #adds item in the ordered files in a directory to all the ordered
```

```
340:            #files
341:            for item in ordered_directory:
342:                ordered_files.append(item)
343:
344:
345:        #adds items in ordered_files to the final print order if the
346:        #extension is .ps
347:        for item in ordered_files:
348:            _, extension = os.path.splitext(item)
349:            if extension == ".ps":
350:                self.print_order.append(item)
351:
352:
353:
354:    def order(self):
355:        """
356:        orders code based on "config.txt" with all the parameters set in
357:        it
358:
359:        @params = None
360:        @return = type list of files to print in order
361:        """
362:        #reset attributes
363:        self.directory_tree = treelib.Tree() #will contain all directories ordered according
364:                                             #to "config.txt"
365:        self.swap_tree = treelib.Tree() #used to re-arrange directories in
366:                                        #other node
367:        self.headers_only = 0 #option set in "config.txt"
368:        self.headers_first = 1 #option set in "config.txt"
369:        self.dir_order = [] #will contain order of directories set in "config.txt"
370:        self.file_order = [] #will contain order of files set in "config.txt"
371:        self.print_order = [] #will contain paths of all print items in order
372:
373:        #order code functions
374:        self.__get_rules()
375:        self.directory_tree, _ = self.return_directory_tree(self.parent)
376:
377:        self.__order_directories()
378:        self.__order_files()
379:
380:        return self.print_order
381:
382:
383:
384:
385:    def print_code(self, files=None):
386:        """
387:        prints code by file and also barriers if they are in the list
388:
389:        @params = (optional) type list of path of files to print
390:        @return = None
391:        """
392:
393:        if files:
394:            self.print_order = files
395:
396:        for file in self.print_order: #adds print job to the printer queue
397:                                      #with shell command
398:            print("queuing ", file)
399:            command = "lpr -P HP-Color-LaserJet-M553 -o sides=two-sided-long-edge " + str(file)
400:            os.system(command)
```

```
#contains rules for print order
#all files are printed alphabetically unless otherwise specified

#parent will specify which directory to start at. It should be set to where pictures are
parent = ../Pictures


#for rules: algorithm first finds all directories and sorts them according
#to rules. Directory rules starts with "dir" (can be in any order).

#Once all directories and their subdirectories are sorted, algorithm
#will look for individual files rules specifies with "file"

#formatting will look best as a directory tree
#only one rule per line

#order for print goes directory, files in that directory, subdirectory,
#files in the subdirectories of that parent


#set to "1" if user wants headers to be printed before implementation files
HEADERS_FIRST 1

#set to "1" if user only wants header files to be printed to save paper
#this will only keep implementation files that have no header
ONLY_HEADERS 0


#rules

#for rules involving headers and implementation files
#specify both the HPP and CPP file
#by setting HEADERS_FIRST the HPP file will be printed first
#even if the CPP file is listed first

dir Print
dir PIDDebugging
    file PIDDebugging/mainPY.ps
dir Prototypes
dir DocumentationMacros
dir CodeGenerator
    file CodeGenerator/code_genPY.ps
    file CodeGenerator/cpp_typesPY.ps
    file CodeGenerator/exceptionsPY.ps
    file CodeGenerator/configPY.ps
dir AutonSimulator
    file AutonSimulator/mainPY.ps
dir Scouting
    file Scouting/ReadmeMD.ps
dir RobotCode
    file RobotCode/lcdTestSH.ps
    file RobotCode/configJSON.ps
    file RobotCode/stacktraceSH.ps
    dir RobotCode/include
    dir RobotCode/src
        file RobotCode/src/mainCPP.ps
        dir RobotCode/src/objects/controller
        dir RobotCode/src/objects/motors
        dir RobotCode/src/objects/sensors
        dir RobotCode/src/objects/robotChassis
        dir RobotCode/src/objects/lift
        dir RobotCode/src/objects/tilter
        dir RobotCode/src/objects/writer
        dir RobotCode/src/objects/lcdCode
            file RobotCode/src/objects/lcdCode/Debug/DebugHPP.ps
            file RobotCode/src/objects/lcdCode/Debug/DebugCPP.ps
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Jun 11 14:26:00 2019

@author: aiden
"""

import copy
import os

import PostScript
import Print
import DirectoryTree


print("starting print process")
print("please be very careful because there is a lot of code")
print()


help_dict = {
    "help":"print help",
    "postscript":"convert files to postscript",
    "page_count":"prints amount of pages that code will be",
    "order":"prepare order to print in",
    "show_order":"shows current order of files to be printed in",
    "tree":"add cover sheet of tree to files",
    "print":"prints code",
    "cancel":"cancels all printer jobs",
    "status":"shows current print jobs",
    "exit":"ends session"
    }


ps = PostScript.PostScript()
printer = Print.Print()
Tree = DirectoryTree.DirectoryTree()


print_order = []


while 1:
    command = input("enter command ")

    if command.upper() == "HELP":
        print()

        for item in help_dict:
            string = item
            while len(string) < 20: #aligns help options with each other
                string = string + "."
            print(string, help_dict.get(item), sep=".")


    elif command.upper() == "POSTSCRIPT":
        ps.prepare_code()

    elif command.upper() == "PAGE_COUNT":
        if not print_order:
            pages = ps.get_page_count()
        else:
            pages = ps.get_page_count(print_order)

        print("number of pages: ", pages)

    elif command.upper() == "ORDER":
        if len(print_order) > 1:
            print_order = []

        to_add = printer.order()

        for item in to_add:
            print_order.append(item)
        print()
        print("please review this order carefully before printing")
        print("do not waste paper")
        print()

        for item in print_order:
            print(item)


    elif command.upper() == "SHOW_ORDER":
        for item in print_order:
            print(item)


    elif command.upper() == "TREE":
        file_tree, _ = Tree.return_tree(".")
        filtered_tree = copy.deepcopy(file_tree)

        #sorts out nodes that are not being printed in three ways
        #Tree is copies to a new tree so that the tree does not
        #change during iteration
        #original is looked through and changes are made to the copy
        for node in file_tree.all_nodes():
            nid = node.identifier
            if os.path.isdir(nid):  #sort out nodes by directory
                if nid in ps.directory_exceptions and filtered_tree.get_node(nid):
                    filtered_tree.remove_node(nid)
            else:
                _, extension = os.path.splitext(nid)

                #sort out nodes by file
                if nid in ps.file_exceptions and filtered_tree.get_node(nid):
                    filtered_tree.remove_node(nid)

                #sort out nodes by extension
                elif extension not in ps.valid_extensions and filtered_tree.get_node(nid):
                    filtered_tree.remove_node(nid)

```

```
114:         filtered_tree.get_node("..").tag = "VexCode-2019"
115:         filtered_tree.show()
116:         os.remove("tree.rtf")
117:
118:         filtered_tree.save2file("tree.rtf")
119:         filtered_tree.save2file("tree.txt")
120:         print_order.insert(0, "tree.rtf")
121:
122:
123:     elif command.upper() == "PRINT":
124:         printer.print_code(print_order)
125:
126:     elif command.upper() == "CANCEL":
127:         os.system("lprm -P HP-Color-LaserJet-M553 -")
128:
129:     elif command.upper() == "STATUS":
130:         os.system("lpstat -P HP-Color-LaserJet-M553")
131:
132:     elif command.upper() == "EXIT":
133:         break
134:
135:     else:
136:         print("invalid command")
137:
138:     print()
139:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Jan  5 17:13:31 2020

@author: aiden
"""
import sys

import parser
import graph

if len(sys.argv) == 1:
    file = input("enter file to parse: ")
else:
    file = sys.argv[1]

parser.gen_sample_data()
p = parser.Parser()
p.parse_file(file)

g = graph.DebugGraph(
    p.get_data()["time"],
    p.get_data()["velocity"],
    p.get_data()["voltage"],
    p.get_data()["setpoint"],
    {
        "kP":p.get_data()["pid_constants"]["kP"],
        "kI":p.get_data()["pid_constants"]["kI"],
        "kD":p.get_data()["pid_constants"]["kD"],
        "I_max":p.get_data()["pid_constants"]["I_max"],
        "brakemode":p.get_data()["brakemode"],
        "gearset":p.get_data()["gearset"],
        "slew":p.get_data()["slew_rate"]
    }
    )

g.get_graph().show()
g.get_graph().savefig("test2.png", bbox_inches='tight')

```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 29 12:38:16 2019

@author: aiden
"""
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D


class DebugGraph:
    """
    class for making a graph for debugging PID data
    """
    def __init__(self, time_data, velocity_data, voltage_data, setpoint, parameters):

#       #calculate axes and round to the nearest 50 that has the higher absolute value
#       time_max = (abs(max(time_data)) / max(time_data)) * 50 * math.ceil(abs(max(time_data))/50)
#
#       velocity_min = (abs(min(velocity_data)) / min(velocity_data)) * 50 * math.ceil(abs(min(velocity_data))/50)
#       velocity_max = (abs(max(velocity_data)) / max(velocity_data)) * 50 * math.ceil(abs(max(velocity_data))/50)
#
#       voltage_min = (abs(min(voltage_data)) / min(voltage_data)) * 50 * math.ceil(abs(min(voltage_data))/50)
#       voltage_max = (abs(max(voltage_data)) / max(voltage_data)) * 50 * math.ceil(abs(max(voltage_data))/50)
#
        #set up graph
        self.__graph = plt.figure(dpi=800)
        self.__ax_vol = self.__graph.add_subplot(111)
        self.__ax_vel = self.__ax_vol.twinx()

        #set up axes
#       self.__ax_vel.set_xlim(0, time_max)
#       self.__ax_vel.set_ylim(velocity_min, velocity_max)
#
#       self.__ax_vol.set_xlim(0, time_max)
#       self.__ax_vol.set_ylim(voltage_min, voltage_max)

        #plot lines
        if type(setpoint) != list:
            setpoint_x = [0, max(time_data)]
            setpoint_y = [setpoint, setpoint]
        else:
            setpoint_x = setpoint
            setpoint_y = time_data

        self.__voltage_line = self.__ax_vol.plot(time_data, voltage_data, color="green", alpha=.25)
        self.__setpoint_line = self.__ax_vel.plot(setpoint_y, setpoint_x, color="blue", linestyle=":")
        self.__velocity_line = self.__ax_vel.plot(time_data, velocity_data, color="blue")

        #set up axis titles
        self.__ax_vel.set_ylabel("Velocity (RPM)")
        self.__ax_vol.set_ylabel("Voltage (mV)")
        self.__ax_vol.set_xlabel("Time (ms)")

        #set axis default colors
        for tl in self.__ax_vel.get_yticklabels():
            tl.set_color("blue")

        for tl in self.__ax_vol.get_yticklabels():
            tl.set_color("green")

        #add legend for lines
        self.__graph.legend(
            [
                Line2D([0], [0], color="blue"),
                Line2D([0], [0], color='blue', linestyle=":"),
                Line2D([0], [0], color="green")
            ],
            ["Actual Velocity", "Setpoint", "Actual Voltage"],
            loc=7, bbox_to_anchor=(1.4, .75))
        self.__graph.subplots_adjust(right=1)

        #add legend for constants
        constants_text = "kP : " + str(parameters.get("kP")) + "\n"
        constants_text += "kI : " + str(parameters.get("kI")) + "\n"
        constants_text += "kD: " + str(parameters.get("kD")) + "\n"
        constants_text += "Integral Max: " + str(parameters.get("I_max")) + "\n"
        constants_text += "\n"
        constants_text += "Brakemode: " + parameters.get("brakemode") + "\n"
        constants_text += "Gearset: " + parameters.get("gearset") + "\n"
        constants_text += "Slew Rate (mV/ms): " + str(parameters.get("slew")) + "\n"

        self.__graph.text(
            1.13,
            .25,
            constants_text,
            )
        self.__graph.subplots_adjust(right=1)


        #add title
        self.__graph.suptitle("Velocity and Voltage vs Time - PID Tuning")



    def set_velocity_color(self, color):
        """
        sets the color of the velocity line on the graph
        """
        self.__setpoint_line.set_color(color)
        self.__velocity_line.set_color(color)

        for tl in self.__ax_vel.get_yticklabels():
            tl.set_color(color)


    def set_voltage_color(self, color):
        """
        sets the color of the voltage line on the graph
        """
        self.__voltage_line.set_color(color)

```

```
114:         for tl in self.__ax_vol.get_yticklabels():
115:             tl.set_color(color)
116:
117:
118:     def get_graph(self):
119:         """
120:         returns the graph object so that it can be saved or manipulated
121:         """
122:         return self.__graph
123:
124:
125: #unit test
126: #
127: #import numpy as np
128: #
129: #time = range(50)
130: #velocity = [20 * np.sin(x) for x in time]
131: #voltage = [x * 60 for x in velocity]
132: #pid = {"kP":1.0,"kI":.001,"kD":.25,"kI_max":5,"brakemode":"Brake","gearset":"18:1","slew":400}
133: #x = DebugGraph(time, velocity, voltage, 12, pid)
134: #x.get_graph().show()
135: #x.get_graph().savefig("test.png", bbox_inches='tight')
136:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jan  6 22:38:31 2020

@author: aiden
"""
class LCD:
    """
    class for communicating with the vex lcd via serial communication
    """
    def __init__(self, serial_write, serial_read):
        self.lcd_screen = serial_write
        self.lcd_buttons = serial_read

        self.__flags = 0x00
        self.__line_1 = " " * 16
        self.__line_2 = " " * 16

        self.__num_chars = 16


    def __write_line(self):
        """
        Writes to the lcd. Runs from thread, not called by user

        Returns
        -------
        int
            1 on success.

        """
        #line 1 bytearray
        send_array = bytearray()
        send_array.append(0xAA)
        send_array.append(0x55)
        send_array.append(0x1E)
        send_array.append(0x12)
        send_array.append(0x00)
        checksum = 0x00

        for char in self.__line_1:
            send_array.append(ord(char))
            checksum += (ord(char))

        self.lcd_screen.write(send_array)

        #line 2 bytearray
        send_array = bytearray()
        send_array.append(0xAA)
        send_array.append(0x55)
        send_array.append(0x1E)
        send_array.append(0x12)
        send_array.append(0x01)
        checksum = 0x01

        for char in self.__line_1:
            send_array.append(ord(char))
            checksum += (ord(char))

        self.lcd_screen.write(send_array)

        return 1



    def write_string(self, string, ln=0, align="left"):
        """
        writes a string to the lcd
        handles multiline writing by keeping track of line one and line two
        and adding a newline character between them

        returns 0 on unsucessful write and 1 if completed successfully
        """
        buffer = self.__num_chars - len(string)
        if align == "center":
            left_spaces = round(buffer/2, 0)
            right_spaces = self.__num_chars - len(string) - left_spaces
            if right_spaces < 0:
                right_spaces = 0
            string = (" " * left_spaces) + string + (" " * right_spaces)

        elif align == "right":
            left_spaces = self.__num_chars - len(string)
            if left_spaces < 0:
                left_spaces = 0
            string = (" " * left_spaces) + string

        elif align == "left":
            string = string

        else:
            return 0

        if len(string) > self.__num_chars:  #cap string length to the number of characters on the lcd
            string = string[:self.__num_chars]


        if ln == 0:
            self.__line_1 = string
        elif ln == 1:
            self.__line_2 = string

        return 1



    def clear(self, line):
        """
        clears a line on the lcd by sending spaces

```

```
114:        Parameters
115:        ----------
116:        line : int
117:            the line to clear.
118:
119:        Returns
120:        -------
121:        int
122:            1 on success, 0 on failure.
123:
124:        """
125:        string = " " * 16
126:        ret = self.write_string(string, ln=line)
127:
128:        if not ret:
129:            return 0
130:
131:        return 1
132:
133:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Jan  5 15:35:01 2020

@author: aiden
"""
import math
import random
import os


class Parser:
    """
    parses data from motors so that it can be graphed
    """
    def __init__(self):
        self.__voltage_data = []
        self.__velocity_data = []
        self.__time_data = []
        self.__integral_data = []
        self.__setpoint_data = []

        self.__brakemode = None
        self.__gearset = None
        self.__slew = 0
        self.__pid = {
            "kP":0,
            "kI":0,
            "kD":0,
            "I_max":0
            }

        self.__brakemode_names = {
            0:"Coast",
            1:"Brake",
            2:"Hold",

        }

        self.__gearset_names = {
            0:"36:1",
            1:"18:1",
            2:"6:1"
        }


    def __get_data_point(self, line):
        """
        parses a line from the file and returns the data of voltage,
        velocity, integral value, and time in the form of a dictionary

        sample line:
            [INFO] Motor 1, Brakemode: xxxx, Actual_Voltage: xxx, ...
        """
        try:
            line = line.split("[INFO]")[1]
            data = line.split(",")
        except IndexError:  #ensures that line is an actual data line
            return 0

        try:
            voltage = [ item.strip().split("Actual_Vol:")[1] for item in data if "Actual_Vol:" in item ]
            velocity = [ item.strip().split("Vel:")[1] for item in data if "Vel:" in item ]
            time = [ item.strip().split("Time:")[1] for item in data if "Time:" in item ]
            integral = [ item.strip().split("I:")[1] for item in data if "I:" in item ]
            setpoint = [ item.strip().split("Vel_Sp:")[1] for item in data if "Vel_Sp:" in item ]

            data_dict = {
                "voltage":float(voltage[0].strip()),
                "velocity":float(velocity[0].strip()),
                "time":float(time[0].strip()),
                "integral":float(integral[0].strip()),
                "setpoint":float(setpoint[0].strip())
                }

            return data_dict
        except IndexError:
            return 0


    def parse_file(self, file):
        """
        parses a file line by line and adds data to list
        """
        f = open(file)

        #find first valid line
        while 1:
            first_line = f.readline()
            data = self.__get_data_point(first_line)

            if data:
                self.__voltage_data.append(data.get("voltage"))
                self.__velocity_data.append(data.get("velocity"))
                self.__time_data.append(data.get("time"))
                self.__integral_data.append(data.get("integral"))
                self.__setpoint_data.append(data.get("setpoint"))

                first_line = first_line.split("[INFO]")[1]
                data = first_line.split(",")

                self.__brakemode = int([ item.strip().split("Brake:")[1].strip() for item in data if "Brake:" in item ][0])
                self.__gearset = int([ item.strip().split("Gear:")[1].strip() for item in data if "Gear:" in item ][0])
                self.__slew = int([ int(item.strip().split("Slew:")[1].strip()) for item in data if "Slew:" in item ][0])

                self.__brakemode = self.__brakemode_names.get(self.__brakemode, "???")
                self.__gearset = self.__gearset_names.get(self.__gearset, "???")

                self.__pid["kP"] = float([ float(item.strip().split("kP:")[1].strip()) for item in data if "kP:" in item ][0])
                self.__pid["kI"] = float([ float(item.strip().split("kI:")[1].strip()) for item in data if "kI:" in item ][0])
                self.__pid["kD"] = float([ float(item.strip().split("kD:")[1].strip()) for item in data if "kD:" in item ][0])
                self.__pid["I_max"] = float([ float(item.strip().split("I_max:")[1].strip()) for item in data if "I_max:" in item ][0])
```

```
114:
115:                 break
116:
117:
118:         for line in f.readlines():
119:             data = self.__get_data_point(line)
120:             if data:
121:                 self.__voltage_data.append(data.get("voltage"))
122:                 self.__velocity_data.append(data.get("velocity"))
123:                 self.__time_data.append(data.get("time"))
124:                 self.__integral_data.append(data.get("integral"))
125:                 self.__setpoint_data.append(data.get("setpoint"))
126:
127:         f.close()
128:
129:
130:     def print_data(self):
131:         """
132:         prints data
133:
134:         useful for debugging
135:         """
136:         print("\nVoltage Data:", len(self.__voltage_data), "data points")
137:         for item in self.__voltage_data:
138:             print(item)
139:
140:         print("\nVelocity Data:", len(self.__velocity_data), "data points")
141:         for item in self.__velocity_data:
142:             print(item)
143:
144:         print("\nIntegral Data:", len(self.__integral_data), "data points")
145:         for item in self.__integral_data:
146:             print(item)
147:
148:
149:         print("\nTime Data:", len(self.__time_data), "data points")
150:         for item in self.__time_data:
151:             print(item)
152:
153:         print("\nSetpoint Data:", len(self.__time_data), "data points")
154:         for item in self.__setpoint_data:
155:             print(item)
156:
157:         print("\nPID constants:")
158:         for key in self.__pid:
159:             print(key, ":", self.__pid[key])
160:
161:
162:         print("\nBrakemode: ", self.__brakemode)
163:         print("Gearset: ", self.__gearset)
164:         print("Slew Rate: ", self.__slew)
165:
166:
167:     def get_data(self):
168:         """
169:         returns a dictionary of the data that was parsed
170:         """
171:         data = {
172:             "voltage":self.__voltage_data,
173:             "velocity":self.__velocity_data,
174:             "integral":self.__integral_data,
175:             "setpoint":self.__setpoint_data,
176:             "time":self.__time_data,
177:             "brakemode":self.__brakemode,
178:             "gearset":self.__gearset,
179:             "slew_rate":self.__slew,
180:             "pid_constants":self.__pid
181:         }
182:
183:         return data
184:
185:
186:
187: def gen_sample_data(num_data_pts=1000, setpoint=100, file="ut.txt"):
188:     """
189:     makes a random set of data that can be used for a unit test
190:     """
191:     os.remove(file)
192:     f = open(file, "a")
193:
194:     step = setpoint / num_data_pts #setpoint /...
195:     min_vel = 0
196:     for i in range(num_data_pts):
197:         vel = (random.randint(int(min_vel), int(min_vel + step)) ** 1/((i+setpoint)/num_data_pts)) + setpoint/10
198:         vol = ((((vel + 200) * (12000 + 12000)) / (200 + 200)) - 12000) + random.randint(-600, 600); #scale vel to voltage range and add jitter
199:         data = "[INFO] Motor 1, Actual_Vol: " + str(vol)
200:         data += ", Brake: 1, Gear: 1, I_max: 1000, I: 100, kD: 0, kI: 0, "
201:         data += "kP: 1.0, Slew: 40, Time: " + str(i)
202:         data += ", Vel_Sp: " + str(setpoint) + ", Vel: " + str(vel) + "\n"
203:
204:         f.write(data)
205:
206:         min_vel += step
207:
208:
209:     f.close()
210:
211: #unit test
212: #
213: #gen_sample_data(file="test.txt")
214: #P = Parser()
215: #P.parse_file("test.txt")
216: #P.print_data()
217:
218:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 13 21:14:17 2019

@author: aiden
"""

import time
import pyautogui

time.sleep(.5)

pyautogui.typewrite("/**")
pyautogui.press("enter")

pyautogui.typewrite(" * @param:")
pyautogui.press("enter")

pyautogui.typewrite("* @param:")
pyautogui.press("enter")

pyautogui.typewrite("* @return:")
pyautogui.press("enter")

pyautogui.typewrite("**")
pyautogui.press("enter")

pyautogui.typewrite("* @see:")
pyautogui.press("enter")

pyautogui.typewrite("* @see:")
pyautogui.press("enter")

pyautogui.typewrite("**")
pyautogui.press("enter")

pyautogui.typewrite("* description of function line 1")
pyautogui.press("enter")

pyautogui.typewrite("* description of function line 2")
pyautogui.press("enter")

pyautogui.typewrite("* description of function line 3")
pyautogui.press("enter")

pyautogui.typewrite("**")
pyautogui.press("enter")

pyautogui.typewrite("*/")
pyautogui.press("enter")

pyautogui.press("backspace")
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 13 21:20:16 2019

@author: aiden
"""

import time
import pyautogui

time.sleep(.5)

pyautogui.typewrite("/**")
pyautogui.press("enter")

pyautogui.typewrite(" * how function works line 1")
pyautogui.press("enter")

pyautogui.typewrite("* how function works line 2")
pyautogui.press("enter")

pyautogui.typewrite("* how function works line 3")
pyautogui.press("enter")


pyautogui.typewrite("*/")
pyautogui.press("enter")

pyautogui.press("backspace")
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 13 21:23:07 2019

@author: aiden
"""

import time
import pyautogui

time.sleep(.5)

pyautogui.typewrite("/**")
pyautogui.press("enter")

pyautogui.typewrite(" * @see:")
pyautogui.press("enter")

pyautogui.typewrite("* @see:")
pyautogui.press("enter")

pyautogui.typewrite("*")
pyautogui.press("enter")

pyautogui.typewrite("* purpose of class line 1")
pyautogui.press("enter")

pyautogui.typewrite("* purpose of class line 2")
pyautogui.press("enter")

pyautogui.typewrite("* purpose of class line 3")
pyautogui.press("enter")

pyautogui.typewrite("*/")
pyautogui.press("enter")

pyautogui.press("backspace")
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Oct 13 20:50:03 2019

@author: aiden
"""

import time
import pyautogui

time.sleep(.5)

pyautogui.typewrite("/**")
pyautogui.press("enter")

pyautogui.typewrite(" * @file:")
pyautogui.press("enter")

pyautogui.typewrite("* @author:")
pyautogui.press("enter")

pyautogui.typewrite("* @reviewed_on:")
pyautogui.press("enter")

pyautogui.typewrite("* @reviewed_by:")
pyautogui.press("enter")

pyautogui.typewrite("* TODO:")
pyautogui.press("enter")

pyautogui.typewrite("*")
pyautogui.press("enter")

pyautogui.typewrite("* description of contents line 1")
pyautogui.press("enter")

pyautogui.typewrite("* description of contents line 2")
pyautogui.press("enter")

pyautogui.typewrite("* description of contents line 3")
pyautogui.press("enter")

pyautogui.typewrite("*")
pyautogui.press("enter")

pyautogui.typewrite("*/")
pyautogui.press("enter")

pyautogui.press("backspace")
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Aug 15 10:16:03 2019

@author: aiden
"""
import inspect
import colorama
import fcntl
import os
import readline
import struct
import termios

import cpp_types
import config
import exceptions


class HeaderGen:
    """
    Wrapper class for working with cpp types and header files
    used to easily generate code based on user input
    """
    def __init__(self, header_obj):
        self.header = header_obj
        self.children = header_obj.get_children()
        self.focus = header_obj
        self.current_type = "header"
        self.loc = "./" + self.header.file_name

        self.commands = {
            "ls":self.__ls,
            "view":self.__view,
            "cd":self.__change_focus,
            "exit":self.__exit,
            "new":self.__new,
            "write":self.__write,
            "add":self.__add,
            "help":self.__help
            }


    @classmethod
    def __exit(cls, *_):
        """
        exit function for shell
        """
        raise exceptions.Exit

    def __ls(self, *_):
        """
        lists data on focused object
        """
        print(self.focus.list_data() + "\n")

    def __view(self, *args):
        """
        shows the generated text of the focused object
        param bool header_text sets the view to either
        the generated header text or the generated implementation
        file text
        """
        if args[0] == []:
            header_text = True
        elif str(args[0][0]).upper() == "HEADER":
            header_text = True
        elif str(args[0][0]).upper() == "IMPLEMENTATION":
            header_text = False
        else:
            raise exceptions.UnknownOption

        text = self.focus.gen_header_text() if header_text else self.focus.gen_impl_text()
        print(text + "\n")

    def __change_focus(self, *args):
        """
        changes focus to user specified input
        no return
        """
        name = args[0][0]
        if name == "..":
            self.focus = self.focus.parent
            self.update_type(self.focus)
            self.children = self.focus.get_children()
        elif self.focus.has_children:
            if name in self.children.keys():
                self.focus = self.children.get(name)
                self.update_type(self.focus)
                self.children = self.focus.get_children()
            else:
                print("invalid selection")
        else:
            print("focused object has no children")


        path = [self.focus]
        while len(path) == len(set(path)):
            path.insert(0, path[0].parent)
        path.pop(0)
        path.pop(0)
        self.loc = "./" + self.header.file_name + "/" + "/".join(x.name for x in path)



    def __new_class(self, name):
        """
        adds a class object to a header file object
        param name = type str of name of the class

        throws invalid addition if not currently focused on header object
        """
```

```
114:         if self.current_type == "header":
115:             obj = cpp_types.cpp_class(name, self.focus)
116:             self.header.classes.append(obj)
117:         else:
118:             raise exceptions.InvalidAddition
119:
120:
121:     def __new_func(self, loc, return_type, name, static=False):
122:         """
123:         adds a function object to a header file or class object
124:         param name = type str of name of the function
125:         param return_type = type of function
126:         param static = is the function static or not
127:
128:         throws invalid addition if incorrect params are passed
129:         """
130:         obj = cpp_types.cpp_func(name, return_type, static, self.focus)
131:
132:         if self.current_type == "header":
133:             self.focus.funcs.append(obj)
134:         elif self.current_type == "class":
135:             if static:
136:                 if loc.upper() == "PUBLIC":
137:                     self.focus.public["static_func"].append(obj)
138:                 elif loc.upper() in ["PROT", "PROTECTED"]:
139:                     self.focus.protected["static_func"].append(obj)
140:                 else:
141:                     self.focus.private["static_func"].append(obj)
142:             else:
143:                 if loc.upper() == "PUBLIC":
144:                     self.focus.public["func"].append(obj)
145:                 elif loc.upper() in ["PROT", "PROTECTED"]:
146:                     self.focus.protected["func"].append(obj)
147:                 else:
148:                     self.focus.private["func"].append(obj)
149:         else:
150:             raise exceptions.InvalidAddition
151:
152:     def __new_var(self, loc, var_type, name, static=False):
153:         """
154:         adds a function object to a header file or class object
155:         param name = type str of name of the function
156:         param var_type = type of variable
157:         param static = is the variable static or not
158:
159:         throws invalid addition if incorrect params are passed
160:         """
161:         obj = cpp_types.cpp_variable(name, var_type, static, self.focus)
162:
163:         if self.current_type == "header":
164:             self.focus.static_vars.append(obj)
165:         elif self.current_type == "class":
166:             if static == "static":
167:                 if loc.upper() == "PUBLIC":
168:                     self.focus.public["static_var"].append(obj)
169:                 elif loc.upper() in ["PROT", "PROTECTED"]:
170:                     self.focus.protected["static_var"].append(obj)
171:                 else:
172:                     self.focus.private["static_var"].append(obj)
173:             else:
174:                 if loc.upper() == "PUBLIC":
175:                     self.focus.public["var"].append(obj)
176:                 elif loc.upper() in ["PROT", "PROTECTED"]:
177:                     self.focus.protected["var"].append(obj)
178:                 else:
179:                     self.focus.private["var"].append(obj)
180:         else:
181:             raise exceptions.InvalidAddition
182:
183:
184:
185:     def __new_include(self, include_type, name):
186:         """
187:         adds an include to a header file
188:         param type (lib, user) - type of include
189:         param name - name of include
190:
191:         throws invalid addition if incorrect params are passed
192:         """
193:         if include_type == "user":
194:             self.header.user_includes.append(name.strip())
195:         elif include_type == "lib":
196:             self.header.lib_includes.append(name.strip())
197:         else:
198:             raise exceptions.InvalidAddition
199:
200:
201:
202:     def __new(self, *args):
203:         """
204:         adds a type of object to another type of object
205:         checks to see if addition is valid
206:         ex. if creating a class the parent must be of type header
207:
208:         throws InvalidAddition if the addition failed
209:         """
210:         func_dict = {
211:             "class":self.__new_class,
212:             "var":self.__new_var,
213:             "func":self.__new_func,
214:             "function":self.__new_func,
215:             "include":self.__new_include
216:             }
217:
218:         if not args[0]:
219:             raise exceptions.InvalidAddition("invalid parameters were passed")
220:
221:         obj_type = args[0][0].strip()
222:         params = list(map(lambda x: x.strip(), args[0][1:]))
223:
224:         func = func_dict.get(obj_type)
225:         if func:
226:             num_args = len(inspect.signature(func).parameters)
```

```
227:            if num_args > len(params):
228:                params.insert(0, "")
229:            while num_args > len(params):
230:                params.append("")
231:
232:        elif func_dict.get(args[0][1].strip()): #if argument in second position
233:                                    #is a valid command then switch
234:                                    #param in first position to front
235:                                    #of params list
236:            func = func_dict.get(args[0][1].strip())
237:            params = list(map(lambda x: x.strip(), args[0][2:]))
238:
239:            num_args = len(inspect.signature(func).parameters)
240:            if num_args > len(params):
241:                params.insert(0, args[0][0].strip())
242:            while num_args > len(params):
243:                params.append("")
244:
245:        else:
246:            raise exceptions.InvalidAddition("invalid function call")
247:
248:        func(*params[:num_args])
249:        self.children = self.focus.get_children()
250:
251:
252:
253:    def __add_function_param(self, param_type, param_name):
254:        """
255:        adds parameters to a function
256:        @param param_type - type str of the cpp type
257:        @param param_name - name of the parameter
258:        throws Invalid Addition on failure or invalid params
259:        """
260:        if self.current_type != "function":
261:            raise exceptions.InvalidAddition
262:
263:        param = param_type.strip() + " " + param_name.strip()
264:        self.focus.params.append(param)
265:
266:
267:    def __add(self, *args):
268:        """
269:        used to add attributes such as parameters to a function
270:
271:        throws invalid addition if the addition failed
272:        """
273:        func_dict = {
274:            "param":self.__add_function_param
275:            }
276:
277:        if not args[0]:
278:            raise exceptions.InvalidAddition("invalid parameters were passed")
279:
280:        obj_type = args[0][0].strip()
281:        params = list(map(lambda x: x.strip(), args[0][1:]))
282:
283:        func = func_dict.get(obj_type)
284:
285:        if func:
286:            num_args = len(inspect.signature(func).parameters)
287:            while num_args > len(params):
288:                params.append("")
289:
290:            func(*params[:num_args])
291:            self.children = self.focus.get_children()
292:
293:        else:
294:            raise exceptions.InvalidAddition("invalid function call")
295:
296:
297:
298:
299:    def __write(self, *_):
300:        """
301:        writes the text generated from the header file into an actual file
302:        as well as generating and writing the text for an implementation
303:        file
304:        """
305:        header_file_name = self.header.file_name
306:        impl_file_name, _ = os.path.splitext(self.header.file_name)
307:        impl_file_name += ".cpp"
308:
309:        with open(header_file_name, "a") as file:
310:            file.write(self.header.gen_header_text())
311:
312:        with open(impl_file_name, "a") as file:
313:            file.write(self.header.gen_impl_text())
314:
315:
316:
317:
318:    def __help(self, *_):
319:        """
320:        prints docstrings for each function
321:        """
322:        #get terminal size to set max chars per line
323:        _, columns, _, _ = struct.unpack('HHHH',
324:                                         fcntl.ioctl(0, termios.TIOCGWINSZ,
325:                                         struct.pack('HHHH', 0, 0, 0, 0)))
326:
327:        max_chars = columns - 5
328:        help_msg = ""
329:        spaces = len(max(list(self.commands.keys()), key=len))
330:
331:        for key in self.commands:
332:            doc_str = str(self.commands.get(key).__doc__).strip().replace("\n", " ")
333:            words = doc_str.split(" ")
334:            words = list(filter(lambda a: a != "", words))
335:
336:            line = key + (" " * (spaces - len(key))) + "  -  "
337:            indentation = len(line) * " "
338:            for word in words:
339:                if (len(word) + len(line)) < max_chars:
```

```
340:                 line += word + " "
341:             else:
342:                 help_msg += line + "\n"
343:                 line = indentation + word + " "
344:         help_msg += line + "\n\n"
345:
346:     print(help_msg)
347:
348:
349:
350:
351:   def update_type(self, obj):
352:     """
353:     updates self.current_type to the type of obj
354:     """
355:     types = {cpp_types.cpp_class:"class",
356:              cpp_types.cpp_func:"function",
357:              cpp_types.HeaderFile:"header",
358:              cpp_types.cpp_variable:"variable"
359:             }
360:     self.current_type = types.get(type(obj))
361:
362:
363:
364:
365:   def execute_command(self, command):
366:     """
367:     executes a command from the given api
368:     api commands are stored in self.commands
369:     """
370:     cmd = self.commands.get(command.split(" ")[0].strip())
371:     args = command.split(" ")[1:]
372:     for arg in args:
373:       arg.strip()
374:
375:     try:
376:       if not cmd:
377:         raise exceptions.UnknownOption
378:       cmd(args)
379:     except exceptions.UnknownOption:
380:       pass
381:     except exceptions.InvalidAddition:
382:       pass
383:
384:
385:
386: #TODO: add dynamically changing autocomplete
387: class Shell:
388:   """
389:   contains shell like interface for generating code
390:   """
391:   def __init__(self):
392:     self.loc = "./"
393:     self.functions = sorted(["new",
394:                              "class",
395:                              "edit",
396:                              "ls",
397:                              "include",
398:                              "user",
399:                              "lib",
400:                              "func",
401:                              "view",
402:                              "exit",
403:                              "add"])
404:
405:
406:   def auto_complete(self, text, state):
407:     """
408:     function that will attempt to autocomplete user input
409:     on <tab> to s member in self.functions
410:     returns type str of first matched string
411:     """
412:     if state == 0:  # on first trigger, build possible matches
413:       if text:  # cache matches (entries that start with entered text)
414:         matches = [s for s in self.functions if s and s.startswith(text)]
415:       else:  # no text entered, all matches possible
416:         matches = self.functions[:]
417:
418:     # return match indexed by state
419:     try:
420:       return matches[state]
421:     except IndexError:
422:       return None
423:
424:
425:   def get_command(self):
426:     """
427:     gets command from user
428:     returns type str of command
429:     """
430:     print("")
431:
432:     colorama.init()
433:     readline.set_completer(self.auto_complete)
434:     readline.parse_and_bind('tab: complete')
435:
436:     command = input((config.SHELL_COLOR
437:                      + "\n"
438:                      + config.CURSOR_UP_ONE
439:                      + config.ERASE_LINE
440:                      + self.loc + " <command> "
441:                      + colorama.Fore.RESET))
442:     return command
443:
444:
445:
446: #def unit_test():
447: #   h = cpp_types.HeaderFile("MyHeader.hpp")
448: #
449: #   c = cpp_types.cpp_class("MyClass")
450: #   c.protected["func"].append(cpp_types.cpp_func("my_func", "int", 1))
451: #   c.public["var"].append("int x")
452: #   h.static_vars.append("int y")
```

```
453:  #    h.classes.append(c)
454:  #
455:  #    t = h.gen_header_text()
456:  #    print(t)
457:
458:
459:
460:
461:  file_name = input(config.SHELL_COLOR
462:                    + "\n"
463:                    + config.CURSOR_UP_ONE
464:                    + config.ERASE_LINE
465:                    + "Enter name of Header File to Create "
466:                    + colorama.Fore.RESET)
467:
468:  if not any(s in file_name and s[-len(s):] == file_name[-len(s):] for s in ['.hpp', '.h']):
469:      raise exceptions.InvalidFileName
470:
471:
472:
473:  header = cpp_types.HeaderFile(file_name)
474:  s = Shell()
475:  s.loc += file_name + "/"
476:  header_gen = HeaderGen(header)
477:
478:
479:  while 1:
480:      try:
481:          try:
482:              usr_command = s.get_command()
483:          except KeyboardInterrupt:
484:              print()
485:              raise exceptions.Exit
486:
487:          header_gen.execute_command(usr_command)
488:          s.loc = header_gen.loc
489:
490:      except exceptions.Exit:
491:          break
492:
493:
494:
495:  #unit_test()
496:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Aug 17 19:52:41 2019

@author: aiden
"""
import os
import config

class cpp_class:
    """
    contains methods and data for a cpp class
    generates text for header and implementation files
    """
    def __init__(self, name, parent):
        self.name = name

        self.has_children = True
        self.parent = parent

        self.private = {
            "func":[],
            "static_func":[],
            "var":[],
            "static_var":[]
            }
        self.protected = {
            "func":[],
            "static_func":[],
            "var":[],
            "static_var":[]
            }
        self.public = {
            "func":[],
            "static_func":[],
            "var":[],
            "static_var":[]
            }



    @classmethod
    def __keys_empty(cls, section):
        """
        checks if all keys in a dictionary have no value
        returns True if empty, False otherwise
        """
        for key in section:
            if section.get(key):
                return False

        return True




    def list_data(self):
        """
        lists the data in the class by category
        returns type str of data
        """
        text = ""
        sections = [self.private,
                    self.protected,
                    self.public]
        section_names = ["private:\n",
                         "protected:\n",
                         "public:\n"]

        for i, section in enumerate(sections):
            text += section_names[i] + "\tstatic variables\n"
            for static_var in section.get("static_var"):
                text += "\t\t" + static_var.var_type + " " + static_var.name + "\n"

            text += "\tvariables\n"
            for var in section.get("var"):
                text += "\t\t" + var.var_type + " " + var.name + "\n"

            text += "\tstatic functions\n"
            for static_func in section.get("static_func"):
                text += "\t\t" + static_func.type + " " + static_func.name + "\n"

            text += "\tfunctions\n"
            for func in section.get("func"):
                text += "\t\t" + func.type + " " + func.name + "\n"

        return text




    def get_children(self):
        """
        returns a dict of children names and their object
        ex. {name1:obj1,
             name2:obj2,
             name3:obj3,
             ...}
        """
        children = {}
        sections = [self.private, self.protected, self.public]

        for section in sections:
            for static_var in section.get("static_var"):
                children.update({static_var.name:static_var})

            for var in section.get("var"):
                children.update({var.name:var})

            for static_func in section.get("static_func"):
                children.update({static_func.name:static_func})

```

```
114:            for func in section.get("func"):
115:                children.update({func.name:func})
116:
117:        return children
118:
119:
120:    def gen_header_text(self):
121:        """
122:        takes data in the class and generates text for a class in
123:        a header file
124:        returns type str of text
125:        """
126:        text = "class " + self.name + "\n{\n\tprivate:\n"
127:
128:        #TODO: condense, function too long, too many branches
129:        for member in self.private.get("static_func"):
130:            text += "\t\t" + member.gen_header_text()
131:        if self.private.get("static_func"):
132:            text += "\n\n"
133:
134:        for member in self.private.get("func"):
135:            text += "\t\t" + member.gen_header_text()
136:        if self.private.get("func"):
137:            text += "\n\n"
138:
139:        for member in self.private.get("static_var"):
140:            text += "\t\t" + member.gen_header_text()
141:        if self.private.get("static_var"):
142:            text += "\n\n"
143:
144:        for member in self.private.get("var"):
145:            text += "\t\t" + member.gen_header_text()
146:        if self.private.get("var"):
147:            text += "\n\n"
148:
149:
150:        prot_txt = ""
151:        if not self.__keys_empty(self.protected):
152:            #if not empty
153:            prot_txt = "\tprotected:\n"
154:            for member in self.protected.get("static_func"):
155:                prot_txt += "\t\t" + member.gen_header_text()
156:            if self.protected.get("static_func"):
157:                prot_txt += "\n\n"
158:
159:            for member in self.protected.get("func"):
160:                prot_txt += "\t\t" + member.gen_header_text()
161:            if self.protected.get("func"):
162:                prot_txt += "\n\n"
163:
164:            for member in self.protected.get("static_var"):
165:                prot_txt += "\t\t" + member.gen_header_text()
166:            if self.protected.get("static_var"):
167:                prot_txt += "\n\n"
168:
169:            for member in self.protected.get("var"):
170:                prot_txt += "\t\t" + member.gen_header_text()
171:            if self.protected.get("var"):
172:                prot_txt += "\n\n"
173:        elif self.__keys_empty(self.protected) and not config.REMOVE_PROTECTED_IF_EMPTY:
174:            #if empty and dont remove protected
175:            prot_txt = "\tprotected:\n"
176:
177:        text += prot_txt + "\tpublic:\n\t\t" + self.name + "();\n\t\t~" + self.name + "();\n\n"
178:
179:
180:        for member in self.public.get("static_func"):
181:            text += "\t\t" + member.gen_header_text()
182:        if self.public.get("static_func"):
183:            text += "\n\n"
184:
185:        for member in self.public.get("func"):
186:            text += "\t\t" + member.gen_header_text()
187:        if self.public.get("func"):
188:            text += "\n\n"
189:
190:        for member in self.public.get("static_var"):
191:            text += "\t\t" + member.gen_header_text()
192:        if self.public.get("static_var"):
193:            text += "\n\n"
194:
195:        for member in self.public.get("var"):
196:            text += "\t\t" + member.gen_header_text()
197:        if self.public.get("var"):
198:            text += "\n\n"
199:
200:        text += "};\n"
201:
202:        tab = ""
203:        for _ in range(config.TAB_SIZE):
204:            tab += " "
205:        text = text.replace("\t", tab)
206:
207:        return text
208:
209:
210:
211:
212:    def gen_impl_text(self):
213:        """
214:        generates text for a class in an implementation file
215:        returns type str of text
216:        """
217:        text = ""
218:
219:        sections = [self.private, self.protected, self.public]
220:        for section in sections:
221:            for static_var in section.get("static_var"):
222:                text += static_var.gen_impl_text()
223:
224:        text += "\n\n\n" + self.name + "::" + self.name + "( )\n{\n\n}\n"
225:        text += "\n\n\n" + self.name + "::~" + self.name + "( )\n{\n\n}\n\n\n\n\n"
226:
```

```
227:         for section in sections:
228:             for static_func in section.get("static_func"):
229:                 text += static_func.gen_impl_text() + "\n\n\n"
230:
231:             for func in section.get("func"):
232:                 text += func.gen_impl_text() + "\n\n\n"
233:
234:         return text
235:
236:
237:
238:
239: class cpp_func:
240:     """
241:     contains data about a cpp function type
242:     can be either static or not and has methods to
243:     generate text in both header and implementation file
244:     """
245:     def __init__(self, name, return_type, static, parent):
246:         self.type = return_type
247:         self.name = name
248:         self.static = bool(static)
249:
250:         self.has_children = False
251:         self.parent = parent
252:
253:         self.params = []
254:
255:
256:
257:
258:     def get_children(self):
259:         """
260:         returns a dict of children names and their object
261:         ex. {name1:obj1,
262:              name2:obj2,
263:              name3:obj3,
264:              ...}
265:         since there are no accesible children, an empty
266:         list is returned
267:         """
268:         return {}
269:
270:
271:
272:     def list_data(self):
273:         """
274:         lists the data in the function by category
275:         returns type str of data
276:         """
277:         text = "type:\n\t" + self.type + "\nname:\n\t" + self.name + "\nstatic:\n\t"
278:         text += "yes\n" if self.static else "no\n"
279:         text += "parameters:\n"
280:         for param in self.params:
281:             text += "\t" + param + "\n"
282:
283:         return text
284:
285:
286:
287:
288:     def gen_header_text(self):
289:         """
290:         generates text for a function with given dataset to be placed
291:         in a header file
292:         returns type str of text for header file
293:         """
294:         text = ""
295:         if self.static:
296:             text += "static "
297:
298:         text += self.type + " "
299:         text += self.name + "( "
300:         for param in self.params:
301:             text += param + ", "
302:         k = text.rfind(",")
303:         if k > 0:
304:             text = text[:k] + text[k+1:]
305:         text += ");\n"
306:
307:         return text
308:
309:
310:
311:
312:     def gen_impl_text(self, class_name=""):
313:         """
314:         generates text for a function with a given dataset to be placed
315:         in an implementation file
316:         returns type str of text for implementation file
317:         """
318:         text = ""
319:
320:         text += self.type + " "
321:         if class_name:
322:             text += class_name + "::" + self.name + "( "
323:         else:
324:             text += self.name + "( "
325:
326:         for param in self.params:
327:             text += param + ", "
328:         k = text.rfind(",")
329:         if k > 0:
330:             text = text[:k] + text[k+1:]
331:
332:         text += ")\n{\n\n}"
333:
334:         return text
335:
336:
337:
338:
339: class cpp_variable:
```

```
340:     """
341:     contains data about a cpp variable type
342:     can be either static or not and has methods to
343:     generate text in both header and implementation file
344:     """
345:     def __init__(self, name, var_type, static, parent):
346:         self.var_type = var_type
347:         self.name = name
348:         self.static = bool(static)
349:
350:         self.has_children = False
351:         self.parent = parent
352:
353:
354:
355:     def get_children(self):
356:         """
357:         returns a dict of children names and their object
358:         ex. {name1:obj1,
359:              name2:obj2,
360:              name3:obj3,
361:              ...}
362:         since there are no accesible children, an empty
363:         list is returned
364:         """
365:         return {}
366:
367:
368:
369:
370:     def list_data(self):
371:         """
372:         lists the data in the variable by category
373:         returns type str of data
374:         """
375:         text = "type:\n\t" + self.var_type + "\nname:\n\t" + self.name + "\nstatic:\n\t"
376:         text += "yes\n" if self.static else "no\n"
377:
378:         return text
379:
380:
381:
382:
383:     def gen_header_text(self):
384:         """
385:         generates text for a variable with given dataset to be placed
386:         in a header file
387:         returns type str of text for header file
388:         """
389:         text = "static " if self.static else ""
390:
391:         text += self.var_type + " " + self.name + ";"
392:
393:         return text
394:
395:
396:
397:
398:     def gen_impl_text(self):
399:         """
400:         generates text for a variable with a given dataset to be placed
401:         in an implementation file
402:         returns type str of text for implementation file
403:         """
404:         return self.var_type + " " + self.name + " = ;\n"
405:
406:
407:
408:
409:
410: class HeaderFile:
411:     """
412:     contains data for a cpp header file
413:     data can also be used to generate an implementation file
414:     """
415:     def __init__(self, file_name):
416:         self.file_name = file_name
417:
418:         self.parent = self
419:         self.has_children = True
420:
421:         self.user_includes = []
422:         self.lib_includes = []
423:         self.classes = []
424:         self.static_vars = []
425:         self.funcs = []
426:
427:
428:
429:
430:     def list_data(self):
431:         """
432:         lists the data in the header file by category
433:         returns type str of data
434:         """
435:         text = "file name:\n\t" + self.file_name + "\nlibrary includes:\n"
436:         for include in self.lib_includes:
437:             text += "\t" + include + "\n"
438:
439:         text += "user includes:\n"
440:         for include in self.user_includes:
441:             text += "\t" + include + "\n"
442:
443:         text += "classes:\n"
444:         for cpp_cls in self.classes:
445:             text += "\t" + cpp_cls.name + "\n"
446:
447:         text += "variables:\n"
448:         for static_var in self.static_vars:
449:             text += "\t" + static_var.gen_header_text() + "\n"
450:
451:         text += "functions:\n"
452:         for func in self.funcs:
```

```
453:            text += "\t" + func.type + " " + func.name + "\n"
454:
455:        return text
456:
457:
458:
459:
460:
461:    def get_children(self):
462:        """
463:        returns a dict of children names and their object
464:        ex. {name1:obj1,
465:             name2:obj2,
466:             name3:obj3,
467:             ...}
468:        """
469:        children = {}
470:
471:        for cpp_cls in self.classes:
472:            children.update({cpp_cls.name:cpp_cls})
473:
474:        for static_var in self.static_vars:
475:            children.update({static_var.name:static_var})
476:
477:        for func in self.funcs:
478:            children.update({func.name:func})
479:
480:        return children
481:
482:
483:
484:
485:    def gen_header_text(self):
486:        """
487:        generates text for a header file
488:        returns type str of text
489:        """
490:        #add header guards
491:        guard, _ = os.path.splitext(self.file_name)
492:        guard = guard.split("/")[-1]
493:
494:        text = "#ifndef __" + guard.upper() + "_HPP__\n"
495:        text += "#define __" + guard.upper() + "_HPP__\n\n"
496:
497:        for include in sorted(self.lib_includes):
498:            text += "#include <" + include + ">\n"
499:
500:        text += '\n#include "main.h"\n\n'
501:
502:        for include in sorted(self.user_includes):
503:            text += '#include "' + include + '"\n'
504:
505:        text += "\n\n"
506:
507:        for c in self.classes:
508:            text += c.gen_header_text() + "\n\n\n\n"
509:
510:        for func in self.funcs:
511:            text += func.gen_header_text() + "\n"
512:        if self.funcs:
513:            text += "\n\n\n\n"
514:
515:        for var in self.static_vars:
516:            text += var.gen_header_text()
517:        if self.static_vars:
518:            text += "\n\n\n\n"
519:
520:        text += "#endif\n"
521:
522:        return text
523:
524:
525:
526:
527:    def gen_impl_text(self):
528:        """
529:        generates text for an implementation file
530:        returns type str of text
531:        """
532:        text = ""
533:        for include in sorted(self.lib_includes):
534:            text += "#include <" + include + ">\n"
535:
536:        text += '\n#include "main.h"\n\n'
537:
538:        text += '#include "' + self.file_name + '"\n'
539:
540:        for include in sorted(self.user_includes):
541:            text += '#include "' + include + '"\n'
542:
543:        text += "\n\n"
544:
545:        for var in self.static_vars:
546:            text += var.gen_impl_text()
547:        if self.static_vars:
548:            text += "\n\n\n\n"
549:
550:        for c in self.classes:
551:            text += c.gen_impl_text() + "\n\n\n\n"
552:
553:        for func in self.funcs:
554:            text += func.gen_impl_text()
555:        if self.funcs:
556:            text += "\n\n\n\n"
557:
558:        return text
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Aug 20 16:48:22 2019

@author: aiden
"""

class Exit(Exception):
    """
    thrown for graceful program termination
    stack trace might not be shown if raised
    """
    pass

class UnknownOption(Exception):
    """
    thrown when an invalid parameter or and invalid command is given
    """
    def __init__(self):
        msg = "Unknown option given"
        super().__init__(msg)
        print(msg)

class InvalidFileName(Exception):
    """
    exception for and invalid file name given
    """
    def __init__(self):
        msg = ("Invalid file name given:\n"
               + "must end in valid header extension type ex.'.hpp' or '.h'")

        super().__init__(msg)
        print(msg)
        raise Exit

class InvalidAddition(Exception):
    """
    addition of object failed
    """
    def __init__(self, msg=None):
        if not msg:
            msg = ("adding of object failed: check to see that parent is of a valid type\n"
                   + "and that parameters were passed correctly")
        super().__init__(msg)
        print(msg)
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Aug 17 19:53:35 2019

@author: aiden
"""
import colorama

TAB_SIZE = 4
REMOVE_PROTECTED_IF_EMPTY = True
SHELL_COLOR = colorama.Fore.LIGHTGREEN_EX
CURSOR_UP_ONE = '\x1b[1A'
ERASE_LINE = '\x1b[2K'
```

```
1: #!/bin/bash
2: #move this to /bin and give executable permissions
3: #to be able to run python script from anywhere
4: python3 /home/aiden/Documents/VexCode-2019/CodeGen/code_gen.py
```

```
1:  import tkinter as tk
2:
3:  import robot
4:  import field
5:  import fieldObjects
6:  import autonomous
7:  import runningFrame
8:  import robotInfoFrame
9:  import controlPanelFrame
10:
11:
12: #user defines
13: X_RES = 1600
14: Y_RES = 900
15: ROBOT_START_TILE = [0, 1]
16: ROBOT_START = [0, 24]
17: ROBOT_START_ANGLE = 0
18:
19:
20:
21: def motion(event):
22:     x, y = event.x, event.y
23:     corrected_coords = [x - 400, y - 50]
24:     print('{}, {}'.format(bot.inches(corrected_coords[0]), bot.inches(corrected_coords[1])))
25:
26:
27:
28: #sets up and draws field
29: f = field.Field(X_RES, Y_RES)
30: f.drawField()
31: f.drawGrid()
32: f.fieldInfo()
33: field = f.field
34: fieldSize = f.distance
35: robotCoordinates = field[ROBOT_START_TILE[0]][ROBOT_START_TILE[1]].placeRobot([ROBOT_START[0], ROBOT_START[1]])
36:
37: fieldObjects.main(field)
38:
39:
40:
41: #gets canvas and tkinter object
42: canvas = f.canvas
43: master = f.master
44:
45:
46:
47: #sets up GUI further
48: master.title("Autonomous Simulator")
49: master.wm_title("Autonomous Simulator")
50: icon = tk.Image("photo", file="autonSimulator.png")
51: master.tk.call('wm', 'iconphoto', master._w, icon)
52: master.resizable(0, 0)
53:
54:
55: #sets up buttons and labels
56: runningPane = runningFrame.runningFrame(master)
57: robotInfoPane = robotInfoFrame.robotInfoFrame(master)
58: controlPanelPane = controlPanelFrame.controlPanelFrame(master, runningPane)
59:
60: runningPane.placeObjects()
61: robotInfoPane.placeObjects()
62: controlPanelPane.placeObjects()
63:
64:
65:
66: #moves robot to start location
67: bot = robot.robot(fieldSize=fieldSize, tkobj=master, canvas=canvas, controlPanelFrame=controlPanelPane, robotInfoFrame=robotInfoPane)
68: bot.show(angle=ROBOT_START_ANGLE, position=robotCoordinates)
69:
70:
71:
72:
73:
74: #runs autonomous
75: try:
76:     master.bind('<Motion>', motion)
77:
78:     auton = autonomous.auton(bot, master, controlPanelPane, robotInfoPane)
79:     auton.commands()
80:
81: except tk.TclError:
82:     pass
83:
84: finally:
85:     try: #finished all the way through
86:         controlPanelPane.disable()
87:
88:         controlPanelPane.idle()
89:
90:     except tk.TclError: #window closed
91:         print("finished")
92:
93:
94:
95:
96:
97:
98:
99:
100:
101:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import time

class auton:
    """
    contains autonomous commands
    """
    def __init__(self, robotObj, master, controlPanelFrame, robotInfoFrame):
        self.controlPanelFrame = controlPanelFrame
        self.robotInfoFrame = robotInfoFrame
        self.robot = robotObj
        self.master = master

        self.orientationLabelText = self.robotInfoFrame.orientationLabelText
        self.orientationLabelText.set("orientation: " + str(self.robot.orientationDegrees))

        self.controlPanelFrame.runningLabelText.set(self.controlPanelFrame.options.get(str(self.controlPanelFrame.keepRunning)))



    def nextFrame(self, waitTime=0):
        """
        updates tkinter canvas and allows for a wait
        """

        self.master.update()

        timeSlept = 0
        while timeSlept <= waitTime:
            if self.controlPanelFrame.keepRunning:
                time.sleep(0.1)
                timeSlept += 0.1
                self.master.update()
            else: #sets robot to idle
                time.sleep(0.1)
                self.master.update()


    def intakeStart(self, rotations):
        """
        for translator
        """
        pass

    def outakeStart(self, rotations):
        """
        for translator
        """
        pass

    def intakeEnd(self, Time):
        """
        for translator
        """
        pass

    def catapult(self, rotations):
        """
        for translator
        """
        pass

    def capFlipper(self, rotations):
        """
        for translator
        """
        pass


    def commands(self):
        """
        autonomous commands
        """
        # self.nextFrame(2)

        ###################### drive forward and backward ###################
        # #deploy
        # self.robot.forward(200)
        # self.nextFrame(.5)

        # self.robot.backward(430)
        # self.nextFrame(.5)

        # self.robot.forward(500)
        # self.nextFrame(.5)
        ########################################################################


        ########################## red big zone auton ######################
        #intake start
        self.robot.forward(400)
        self.nextFrame(.5)

        self.robot.backward(250)
        self.nextFrame(.5)

        self.robot.forward(450) # move to in front of cube
        self.nextFrame(.5)

        #move arms up

        self.robot.forward(200) # move to in front of cube
        self.nextFrame(.5)

        #move arms down while going forward slowly
        self.robot.forward(350)
        self.nextFrame(.5)

        #back up
        self.robot.backward(630)
        self.nextFrame(.1)
```

```
114:
115:        #turn towards cube
116:        self.robot.turnLeft(90)
117:        self.nextFrame(.5)
118:
119:        #drive up to cube
120:        self.robot.forward(330)
121:        self.nextFrame(.5)
122:
123:        #intake cube
124:        #intake start
125:        self.robot.forward(340)
126:        self.nextFrame(.5)
127:
128:        self.robot.turnLeft(50)
129:        self.nextFrame(.5)
130:
131:        self.robot.forward(430)
132:        self.nextFrame(.5)
133:
134:        #dump stack
135:
136:        ######################################################################
137:
138:
139:
140:    #drive forward and backward
141:        # self.nextFrame(2)
142:
143:        # self.robot.forward(400)
144:        # self.nextFrame(.1)
145:
146:
147:        # self.robot.backward(500)
148:        # self.nextFrame(.1)
149:
150:        ########################## red small zone auton ####################
151:        # self.nextFrame(2)
152:
153:        # self.robot.forward(440) #move to cube
154:        # self.nextFrame(1)
155:
156:        # for i in range(0,4):
157:        #     self.robot.forward(160)
158:        #     self.nextFrame(.2)
159:
160:        # self.robot.forward(50)
161:        # self.nextFrame(.1)
162:
163:        # self.robot.backward(500)
164:        # self.nextFrame(1)
165:
166:        # self.robot.turnRight(180)
167:        # self.nextFrame(1)
168:
169:        # self.robot.forward(400)
170:        # self.nextFrame(1)
171:
172:        # self.robot.turnLeft(65)
173:        # self.nextFrame(1)
174:
175:        # self.robot.forward(300)
176:        # self.nextFrame(1)
177:
178:
179:
180:
181:    ########################## unit test ##########################
182:    #       import random                           #
183:    #       for x in range(0,4):                    #
184:    #           self.robot.forward(500)               #
185:    #           self.nextFrame(1)                   #
186:    #                                           #
187:    #           self.robot.turnLeft(90)             #
188:    #           self.nextFrame(1)                   #
189:    #                                           #
190:    #       self.robot.turnLeft(45)                 #
191:    #       self.nextFrame(1)                       #
192:    #                                           #
193:    #       for x in range(0,4):                    #
194:    #           self.robot.forward(400)               #
195:    #           self.nextFrame(1)                   #
196:    #                                           #
197:    #           self.robot.turnRight(90)              #
198:    #           self.nextFrame(1)                   #
199:    #                                           #
200:    #       self.robot.turnRight(45)                #
201:    #       self.nextFrame(1)                       #
202:    #                                           #
203:    #                                           #
204:    #       for x in range(0, 6):                   #
205:    #           self.robot.forward(500)               #
206:    #           self.nextFrame(1)                   #
207:    #                                           #
208:    #           self.robot.turnRight(random.randint(10,90))    #
209:    #           self.nextFrame(1)                   #
210:    #                                           #
211:    ######################################################################
212:
213:
214:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import smtplib
import getpass


SEND_EMAIL = 0
FILE = 'autonomous.py'


class email():
    def __init__(self):
        self.passwd = None
        self.user_email = 'jg570144@gmail.com'
        self.recipients = []

        self.message = None

        self.smtpObj = None
        self.check = None
        self.subject = None
        self.body = None
        self.sent = 0


    def getCredentials(self):
        #self.user_email = input("email address:\n")
        self.passwd = getpass.getpass()


    def getMessage(self, oldCode, newCode):
        r = int(input("\nEnter number of recipients\n"))

        for x in range(r):
            remail = input("enter recipient email\n")
            self.recipients.append(remail)

        self.subject = "translated autonomous code"

        self.body = "old auton code:\n\n"

        for i in oldCode:
            self.body = self.body + i + '\n'
            print("")

        self.body = self.body + '\n\n\n\nnew auton code:\n\n'

        for i in finalCode:
            self.body = self.body + i + '\n'

        self.body = self.body + '\n\n\nthis is an automated message\ncourtesy of Aiden'
        print("")


    def login(self):
        self.smtpObj = smtplib.SMTP("smtp.gmail.com:587")
        self.smtpObj.starttls()
        self.smtpObj.ehlo()
        self.smtpObj.login(self.user_email, self.passwd)


    def send(self):
        print("sent: ")

        message = "Subject: " + self.subject + "\n" + self.body

        self.check = self.smtpObj.sendmail(self.user_email, self.recipients, message)

    def close(self):
        self.smtpObj.quit()









class Translator:

    def __init__(self, file):
        self.file = open(file)
        self.code = self.file.readlines()

        self.oldCode = []
        self.newCode = []

        self.conversion = {
            "turnRight": "turnRightV",
            "turnLeft": "turnLeftV",
            "rightSide": "rs",
            "leftSide": "ls",
            "forward": "driveForward",
            "backward": "driveForward",
            "reverse": "changeDirection",
            "intakeStart": "intakeStart",
            "intakeEnd": "intakeEnd",
            "outake": "intakeStart",
            "catapult": "shootBall",
            "capFlipper": "flip"
            }


        self.functionMap = {
            "value": self.value,
            "sleep100RPM": self.sleep100RPM,
```

```
114:                "sleep200RPM": self.sleep200RPM,
115:                "sleep600RPM": self.sleep600RPM,
116:            "negativeValue": self.negativeValue,
117:            "noParam": self.noParam
118:             }
119:
120:
121:        self.specialInstructions = {
122:            "forward": ["value", "sleep200RPM"],
123:            "backward": ["negativeValue", "sleep200RPM"],
124:            "turnLeft": ["noParam"],
125:            "turnRight": ["noParam"],
126:            "intakeStart": ["negativeValue"],
127:            "outakeStart": ["value"],
128:            "intakeEnd": ["sleep600"],
129:            "catapult": ["value", "sleep100"],
130:            "capFlipper": ["value"],
131:            }
132:
133:
134:
135:        self.parameterValue = ""
136:        self.sleepTime = ""
137:        self.separator = ""
138:        self.instructions = []
139:
140:
141:    def value(self):
142:        self.parameterValue = str(self.parameterValue)
143:        self.sleepTime = ""
144:        self.separator = ""
145:
146:    def sleep100RPM(self):
147:        RPM = 100
148:        RPS = RPM / 60
149:        revolutions = (abs(float(self.parameterValue)) / 360)
150:        self.sleepTime = (1000 * (revolutions / RPS)) + 50
151:        #self.sleepTime = str(int((1000 * abs((float(self.parameterValue))/360))) + 200)
152:        if len(self.instructions) == 1:
153:            self.parameterValue = ""
154:            self.separator = ""
155:        else:
156:            self.separator = ", "
157:
158:    def sleep200RPM(self):
159:        RPM = 200
160:        RPS = RPM / 60
161:        revolutions = (abs(float(self.parameterValue)) / 360)
162:        self.sleepTime = (1000 * (revolutions / RPS)) + 50
163:        if len(self.instructions) == 1:
164:            self.parameterValue = ""
165:            self.separator = ""
166:        else:
167:            self.separator = ", "
168:
169:    def sleep600RPM(self):
170:        RPM = 600
171:        RPS = RPM / 60
172:        revolutions = (abs(float(self.parameterValue)) / 360)
173:        self.sleepTime = (1000 * (revolutions / RPS)) + 50
174:        if len(self.instructions) == 1:
175:            self.parameterValue = ""
176:            self.separator = ""
177:        else:
178:            self.separator = ", "
179:
180:
181:    def negativeValue(self):
182:        self.parameterValue = str(0 - float(self.parameterValue))
183:        self.sleepTime = ""
184:        self.separator = ""
185:
186:    def noParam(self):
187:        self.parameterValue = ""
188:        self.sleepTime = ""
189:        self.separator = ""
190:
191:
192:
193:
194:
195:    def translate(self):
196:
197:        for i in self.code:
198:
199:            try:
200:                oldCommand = i.split(".")[1]
201:                try:
202:                    rawCommand = i.split(".")[2]
203:                    oldCommand = oldCommand + "." + rawCommand
204:                except:
205:                    rawCommand = i.split(".")[1]
206:
207:                oldCommand = oldCommand.split(")")[0]
208:                oldCommand = 'self.' + oldCommand + ")"
209:
210:                command = rawCommand.split("(")[0]
211:                value = rawCommand.split("(")[1]
212:                self.parameterValue = value.split(")")[0]
213:
214:                self.instructions = list(self.specialInstructions.get(command))
215:                for j in self.instructions:
216:                    self.functionMap[str(j)]()
217:
218:                newCommand = self.conversion.get(command)
219:                newCommand = (newCommand
220:                              + "(" + str(self.parameterValue)
221:                              + str(self.separator)
222:                              + str(self.sleepTime)
223:                              + ");"
224:                              )
225:
226:                self.oldCode.append(oldCommand)
```

```
227:                self.newCode.append(newCommand)
228:                self.parameterValue = ""
229:                self.sleep = ""
230:                self.separator = ""
231:
232:            except:
233:                self.parameterValue = ""
234:                self.sleep = ""
235:                self.separator = ""
236:
237:        return self.oldCode, self.newCode
238:
239:
240:
241:  t = Translator(FILE)
242:  oldCode, finalCode = t.translate()
243:
244:
245:  if SEND_EMAIL:
246:      s = email()
247:      s.getCredentials()
248:      s.getMessage(oldCode, finalCode)
249:      s.login()
250:      s.send()
251:      s.close()
252:
253:      print("")
254:      print("")
255:      print("done")
256:
257:  else:
258:      for i in oldCode:
259:          print(i)
260:      print("")
261:      for i in finalCode:
262:          print(i)
263:
264:
265:
266:
267:
268:
269:
270:
271:
272:
273:
274:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import tkinter as tk
import time

class controlPanelFrame:
    """
    makes frame object for all labels and buttons
    """
    def __init__(self, master, runningFrame):
        self.runningLabelText = runningFrame.runningLabelText
        self.master = master

        self.guiFrame = tk.Frame(self.master)
        self.guiFrame.grid(row=1, column=0, sticky='news')

        self.speed = 7
        self.keepRunning = True

        self.options = {
            "True": "running",
            "False":"paused "
            }

        self.pauseButtonTextOptions = {
            "True": "pause ",
            "False":"resume"
             }


        ##### labels #####

        #speed label
        self.speedLabelText = tk.StringVar()
        self.speedLabel = tk.Label(self.guiFrame, textvariable=self.speedLabelText)


        ##### buttons #####

        self.fasterButton = tk.Button(self.guiFrame, text=' + ', command=self.__faster)
        self.slowerButton = tk.Button(self.guiFrame, text=' - ', command=self.__slower)

        self.pauseButtonText = tk.StringVar()
        self.pauseButton = tk.Button(self.guiFrame, textvariable=self.pauseButtonText, command=self.__pause)
        self.master.bind("<space>", self.__pause)




    def __changeSpeedLabel(self):
        self.speedLabelText.set("speed: " + str(self.speed))



    def __faster(self):
        if self.speed < 10:
            self.speed = self.speed + 1
            self.__changeSpeedLabel()

    def __slower(self):
        if self.speed > 1:
            self.speed = self.speed - 1
            self.__changeSpeedLabel()

    def __pause(self, event=None):
        """
        pauses robot actions
        """
        self.keepRunning = not self.keepRunning

        self.runningLabelText.set(self.options.get(str(self.keepRunning)))
        self.pauseButtonText.set(self.pauseButtonTextOptions.get(str(self.keepRunning)))

        self.master.update()


    def idle(self):
        """
        sends canvas into idle state
        """
        while 1:
            time.sleep(0.1)
            self.master.update()

    def disable(self):
        self.pauseButton.config(state="disabled")
        self.fasterButton.config(state="disabled")
        self.slowerButton.config(state="disabled")

        self.master.unbind("<space>")

    def placeObjects(self):
        """
        places all buttons and labels
        """

        self.pauseButton.grid(row=0, column=2, columnspan=4, rowspan=2, sticky='news', ipadx=30)

        self.speedLabel.grid(row=2, column=2, columnspan=4, rowspan=1, sticky='news')

        self.slowerButton.grid(row=0, column=0, rowspan=3, columnspan=2, sticky='news')
        self.fasterButton.grid(row=0, column=6, rowspan=3, columnspan=2, sticky='news')


        #sets default value of labels
        self.pauseButtonText.set(self.pauseButtonTextOptions.get(str(self.keepRunning)))
        self.speedLabelText.set("speed: " + str(self.speed))
        self.runningLabelText.set(self.options.get(str(self.keepRunning)))
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


def main(field):
    """
    creates field elements
    """
    for num in range(0,6): #draw white line auton line
        field[2][num].drawRectangle(vertexes=[[47,0],[47,47],[47,47],[47,0]], color="#ffffff", width=5, outline="#ffffff")


    #starting spots
    field[0][1].drawObjectTiles(color="red")
    field[1][0].drawObjectTiles(color="red")
    field[4][0].drawObjectTiles(color="blue")
    field[5][1].drawObjectTiles(color="blue")


    #white tape by auton spots
    #horizontal lines
    field[0][0].drawRectangle(vertexes=[[0, 47], [47, 47], [47, 47], [0, 47]], color='white')
    field[0][1].drawRectangle(vertexes=[[0, 47], [47, 47], [47, 47], [0, 47]], color='white')
    field[5][0].drawRectangle(vertexes=[[0, 47], [47, 47], [47, 47], [0, 47]], color='white')
    field[5][1].drawRectangle(vertexes=[[0, 47], [47, 47], [47, 47], [0, 47]], color='white')

    #vertical lines
    field[0][0].drawRectangle(vertexes=[[47, 0], [47, 47], [47, 47], [47, 0]], color='white')
    field[4][0].drawRectangle(vertexes=[[47, 0], [47, 47], [47, 47], [47, 0]], color='white')
    field[1][0].drawRectangle(vertexes=[[47, 0], [47, 47], [47, 47], [47, 0]], color='white')
    field[3][0].drawRectangle(vertexes=[[47, 0], [47, 47], [47, 47], [47, 0]], color='white')

    #angled lines
    field[1][1].drawRectangle(vertexes=[[0, 47], [47, 0], [47, 0], [0, 47]], color='white')
    field[4][1].drawRectangle(vertexes=[[0, 0], [47, 47], [47, 47], [0, 0]], color='white')



    #cubes
    #row 0
    field[1][0].drawObjectFieldElementSquare(position=[5, 42], color='orange', size=5)
    field[2][0].drawObjectFieldElementSquare(position=[47, 7], color='purple', size=5)
    field[4][0].drawObjectFieldElementSquare(position=[42, 42], color='green', size=5)


    #row 1
    field[1][1].drawObjectFieldElementSquare(position=[5, 42], color='orange', size=5)
    field[2][1].drawObjectFieldElementSquare(position=[5, 42], color='orange', size=5)

    field[2][1].drawObjectFieldElementSquare(position=[33, 24], color='orange', size=5)
    field[2][1].drawObjectFieldElementSquare(position=[47, 11], color='purple', size=5)
    field[2][1].drawObjectFieldElementSquare(position=[47, 37], color='purple', size=5)
    field[3][1].drawObjectFieldElementSquare(position=[14, 24], color='green', size=5)


    field[3][1].drawObjectFieldElementSquare(position=[41, 42], color='green', size=5)
    field[4][1].drawObjectFieldElementSquare(position=[41, 42], color='green', size=5)


    #rows 2 and 3
    field[0][2].drawObjectFieldElementSquare(position=[47, 33], color='purple', size=5)
    field[0][3].drawObjectFieldElementSquare(position=[47, 14], color='purple', size=5)
    field[2][2].drawObjectFieldElementSquare(position=[47, 33], color='purple', size=5)
    field[2][3].drawObjectFieldElementSquare(position=[47, 14], color='purple', size=5)
    field[4][2].drawObjectFieldElementSquare(position=[47, 33], color='purple', size=5)
    field[4][3].drawObjectFieldElementSquare(position=[47, 14], color='purple', size=5)

    field[2][2].drawObjectFieldElementSquare(position=[5, 42], color='orange', size=5)
    field[2][3].drawObjectFieldElementSquare(position=[5, 42], color='orange', size=5)
    field[3][2].drawObjectFieldElementSquare(position=[41, 42], color='green', size=5)
    field[3][3].drawObjectFieldElementSquare(position=[41, 42], color='green', size=5)

    field[2][2].drawObjectFieldElementSquare(position=[33, 47], color='orange', size=5)
    field[3][2].drawObjectFieldElementSquare(position=[14, 47], color='green', size=5)


    field[0][2].drawObjectFieldElementSquare(position=[33, 47], color='orange', size=5)
    field[5][2].drawObjectFieldElementSquare(position=[14, 47], color='green', size=5)

    field[1][2].drawObjectFieldElementSquare(position=[41, 42], color='purple', size=5)
    field[4][2].drawObjectFieldElementSquare(position=[6, 42], color='purple', size=5)

    field[1][3].drawObjectFieldElementSquare(position=[41, 42], color='green', size=5)
    field[4][3].drawObjectFieldElementSquare(position=[6, 42], color='orange', size=5)

    field[1][3].drawObjectFieldElementSquare(position=[29, 42], color='purple', size=5)
    field[4][3].drawObjectFieldElementSquare(position=[18, 42], color='purple', size=5)


    #row 4
    field[3][4].drawObjectFieldElementSquare(position=[41, 42], color='green', size=5)
    field[2][4].drawObjectFieldElementSquare(position=[5, 42], color='orange', size=5)
    field[1][4].drawObjectFieldElementSquare(position=[41, 42], color='green', size=5)
    field[4][4].drawObjectFieldElementSquare(position=[6, 42], color='orange', size=5)
    field[1][4].drawObjectFieldElementSquare(position=[29, 42], color='green', size=5)
    field[4][4].drawObjectFieldElementSquare(position=[18, 42], color='orange', size=5)
    field[1][4].drawObjectFieldElementSquare(position=[18, 42], color='purple', size=5)
    field[4][4].drawObjectFieldElementSquare(position=[29, 42], color='purple', size=5)

    field[2][4].drawObjectFieldElementSquare(position=[33, 24], color='orange', size=5)
    field[2][4].drawObjectFieldElementSquare(position=[47, 11], color='purple', size=5)
    field[2][4].drawObjectFieldElementSquare(position=[47, 37], color='purple', size=5)
    field[3][4].drawObjectFieldElementSquare(position=[14, 24], color='green', size=5)


    #row 5
    field[2][5].drawObjectFieldElementSquare(position=[42, 30], color='green', size=5)
    field[2][5].drawObjectFieldElementSquare(position=[42, 40], color='green', size=5)

    field[3][5].drawObjectFieldElementSquare(position=[5, 30], color='orange', size=5)
    field[3][5].drawObjectFieldElementSquare(position=[5, 40], color='orange', size=5)

```

```
114:    field[2][5].drawObjectFieldElementSquare(position=[47, 35], color='purple', size=5)
115:
116:
117:    #towers
118:    field[2][1].drawObjectFieldElementCircle(position=[47, 24], color='#919191')
119:    field[2][2].drawObjectFieldElementCircle(position=[47, 47], color='#919191')
120:    field[2][4].drawObjectFieldElementCircle(position=[47, 24], color='#919191')
121:
122:    field[0][2].drawObjectFieldElementCircle(position=[47, 47], color='#919191')
123:    field[4][2].drawObjectFieldElementCircle(position=[47, 47], color='#919191')
124:
125:    field[1][5].drawObjectFieldElementCircle(position=[0, 47], color='#919191')
126:    field[4][5].drawObjectFieldElementCircle(position=[47, 47], color='#919191')
127:
128:
129:
130:    #goals
131:    field[0][0].drawRectangle(vertexes=[[0, 24], [32, 24], [32, 24], [0, 24]], outline='red', width=6)
132:    field[0][0].drawRectangle(vertexes=[[32, 0], [32, 24], [32, 24], [32, 0]], outline='red', width=6)
133:
134:    field[5][0].drawRectangle(vertexes=[[47, 24], [16, 24], [16, 24], [47, 24]], outline='blue', width=6)
135:    field[5][0].drawRectangle(vertexes=[[16, 0], [16, 24], [16, 24], [16, 0]], outline='blue', width=6)
136:
137:
138:    field[0][5].drawRectangle(vertexes=[[0, 24], [24, 24], [24, 24], [0, 24]], outline='red', width=6)
139:    field[0][5].drawRectangle(vertexes=[[24, 47], [24, 24], [24, 24], [24, 47]], outline='red', width=6)
140:
141:    field[5][5].drawRectangle(vertexes=[[47, 24], [24, 24], [24, 24], [47, 24]], outline='blue', width=6)
142:    field[5][5].drawRectangle(vertexes=[[24, 47], [24, 24], [24, 24], [24, 47]], outline='blue', width=6)
143:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import tkinter as tk


class Tile:
    """
    gives ability to draw objects on each tile
    """
    def __init__(self, x, y, distance, canvas):
        self.canvas = canvas

        self.distance = distance

        self.P1 = [x, y]
        self.P2 = [x + distance, y]
        self.P3 = [x, y + distance]
        self.P4 = [x + distance, y + distance]


        self.grid = []
        x = self.P1[0]
        increment = self.distance/47
        for i in range(0, 48): #makes grid with coordinates
            column = []
            y = self.P1[1]
            for j in range(0, 48):
                coords = [x, y]
                column.append(coords)
                y = y + increment
            x = x + increment
            self.grid.append(column)


    def __makeCenteredGrid(self, size):
        centeredGrid = []
        x = self.P1[0]
        increment = (self.distance - size) / 2
        for i in range(0, 3): #makes grid with coordinates
            column = []
            y = self.P1[1]
            for j in range(0, 3):
                coords = [x, y]
                column.append(coords)
                y = y + increment
            x = x + increment
            centeredGrid.append(column)

        return centeredGrid


    def drawObjectTiles(self, position=[23, 23], size=24, color="#cccccc", outline=None, width=None):
        """
        draws object to fill a square of the grid in inches
        """
        x1 = self.grid[position[0]][position[1]][0]
        y1 = self.grid[position[0]][position[1]][1]

        s = self.distance/25
        for i in range(0, size):
            s = s + (self.distance/25)

        x1 = x1 - (s/2)
        y1 = y1 - (s/2)

        x2 = self.grid[position[0]][position[1]][0] + (s/2)
        y2 = self.grid[position[0]][position[1]][1] + (s/2)

        self.canvas.create_rectangle(x1, y1, x2, y2, outline=outline, fill=color, width=width)


    def drawObjectFieldElementSquare(self, position=[1, 1], size=8, color="#cccccc", outline=None, width=None):
        """
        draws object centered on the coordinate chosen in inches
        """
        x1 = self.grid[position[0]][position[1]][0]
        y1 = self.grid[position[0]][position[1]][1]

        s = self.distance/25
        for i in range(0, size):
            s = s + (self.distance/25)

        x1 = x1 - (s/2)
        y1 = y1 - (s/2)

        x2 = self.grid[position[0]][position[1]][0] + (s/2)
        y2 = self.grid[position[0]][position[1]][1] + (s/2)

        self.canvas.create_rectangle(x1, y1, x2, y2, outline=outline, fill=color, width=width)


    def drawObjectFieldElementCircle(self, position=[1, 1], size=8, color="#cccccc", outline=None, width=None):
        """
        draws object centered on the coordinate chosen in inches
        """
        x1 = self.grid[position[0]][position[1]][0]
        y1 = self.grid[position[0]][position[1]][1]

        s = self.distance/25
        for i in range(0, size):
            s = s + (self.distance/25)

        x1 = x1 - (s/2)
        y1 = y1 - (s/2)

        x2 = self.grid[position[0]][position[1]][0] + (s/2)
        y2 = self.grid[position[0]][position[1]][1] + (s/2)

        self.canvas.create_oval(x1, y1, x2, y2, outline=outline, fill=color, width=width)


```

```
114:
115:
116:    def drawObjectFieldElementCentered(self, position=[1, 1], size=8, color="#cccccc", outline=None, width=None):
117:        """
118:        draws object contained only in tile in inches
119:        """
120:        s = self.distance/25
121:        for i in range(0, size):
122:            s = s + (self.distance/25)
123:
124:        centeredGrid = self.__makeCenteredGrid(s)
125:
126:        x1 = centeredGrid[position[0]][position[0]][0]
127:        y1 = centeredGrid[position[0]][position[1]][1]
128:
129:        x2 = centeredGrid[position[0]][position[0]][0] + s
130:        y2 = centeredGrid[position[0]][position[1]][1] + s
131:
132:        self.canvas.create_rectangle(x1, y1, x2, y2, outline=outline, fill=color, width=width)
133:
134:
135:    def drawRectangle(self, vertexes=[[0, 1], [2, 1], [2, 1], [0, 1]], color="#cccccc", outline="white", width=3):
136:        """
137:        draws a rectangle given four vertices
138:        """
139:        points = [
140:            [
141:                self.grid[vertexes[0][0]][vertexes[0][0]][0],
142:                self.grid[vertexes[0][1]][vertexes[0][1]][1]
143:            ],
144:            [
145:                self.grid[vertexes[1][0]][vertexes[1][0]][0],
146:                self.grid[vertexes[1][1]][vertexes[1][1]][1]
147:            ],
148:            [
149:                self.grid[vertexes[2][0]][vertexes[2][0]][0],
150:                self.grid[vertexes[2][1]][vertexes[2][1]][1]
151:            ],
152:            [
153:                self.grid[vertexes[3][0]][vertexes[3][0]][0],
154:                self.grid[vertexes[3][1]][vertexes[3][1]][1]
155:            ]
156:
157:            ]
158:
159:        self.canvas.create_polygon(points, outline=outline, fill=color, width=width)
160:
161:
162:
163:    def placeRobot(self, position=[1, 1], size=17):
164:        """
165:        draws robot in tile in inches
166:        """
167:        s = self.distance/25
168:        for i in range(0, size):
169:            s = s + (self.distance/25)
170:
171:        #centeredGrid = self.__makeCenteredGrid(s)
172:
173:        x1 = self.grid[position[0]][position[0]][0]
174:        y1 = self.grid[position[0]][position[1]][1]
175:
176:        x2 = self.grid[position[0]][position[0]][0] + s
177:        y2 = self.grid[position[0]][position[1]][1] + s
178:
179:        coords = [[x1, y1], [x2, y2]]
180:        return coords
181:
182:
183:
184:
185:
186:
187:
188:
189:class Field:
190:    """
191:    creates the field
192:    """
193:    def __init__(self, width, height):
194:        master = tk.Tk()
195:        self.master = master
196:        self.canvas = tk.Canvas(master, width=width, height=height)
197:        self.canvas.grid(row=0, column=0, columnspan=3)
198:
199:        self.canvas.create_rectangle(0, 0, width, height, fill="#ffffff")
200:
201:        distance = height - 100
202:        self.P1 = [((width-distance)/2), 50]
203:        self.P2 = [(((width-distance)/2) + (distance)), 50]
204:        self.P3 = [((width-distance)/2), (50+distance)]
205:        self.P4 = [(((width-distance)/2) + (distance)), (50+distance)]
206:
207:        self.width = width
208:        self.height = height
209:        self.distance = distance
210:
211:        self.field = []
212:
213:
214:
215:
216:    def drawField(self):
217:        """
218:        draws the field based on the resolution given
219:        """
220:        self.canvas.create_rectangle(self.P1[0], self.P1[1], self.P4[0], self.P4[1], fill="#cccccc")
221:
222:        self.canvas.create_line(self.P1[0], self.P1[1], self.P2[0], self.P2[1], fill="black", width=7) #horizontal
223:        self.canvas.create_line(self.P1[0], self.P1[1] - 3, self.P3[0], self.P3[1] + 4, fill="black", width=7) #vertical
224:
225:        self.canvas.create_line(self.P2[0], self.P2[1] - 3, self.P4[0], self.P4[1] + 4, fill="black", width=7) #vertical
226:        self.canvas.create_line(self.P3[0], self.P3[1], self.P4[0], self.P4[1], fill="black", width=7) #horizontal
```

```
227:
228:
229:
230:     def drawGrid(self):
231:         """
232:         draws grid on the field
233:         """
234:         start = (self.width - self.distance) / 2
235:         increment = self.distance / 6
236:
237:         for x in range(0, 6): #verical lines
238:             P1 = [start + (increment*x), 50]
239:             P2 = [start + (increment*x), (50+self.distance)]
240:
241:             self.canvas.create_line(P1[0], P1[1], P2[0], P2[1], fill="black", width=2)
242:
243:         start = 50
244:         for x in range(0, 6): #horizontal lines
245:             P1 = [((self.width - self.distance) / 2), start + (increment*x)]
246:             P2 = [(((self.width - self.distance) / 2) + self.distance), start + (increment*x)]
247:
248:             self.canvas.create_line(P1[0], P1[1], P2[0], P2[1], fill="black", width=2)
249:
250:
251:     def fieldInfo(self):
252:         """
253:         creates objects of each tile and appends them to a list
254:         so that it is possible to create objects on the field
255:         """
256:         distance = self.distance/6
257:         x = ((self.width-self.distance)/2)
258:
259:         for i in range(0, 6): #columns
260:             column = []
261:             y = 50
262:             for j in range(0, 6): #rows
263:                 tileObject = Tile(x, y, distance, self.canvas)
264:                 column.append(tileObject)
265:
266:                 y = y + distance
267:             x = x + distance
268:
269:             self.field.append(column)
270:
271:
272:
273:
274:
275:
276:
277:
278:
279:
280:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import tkinter as tk

class robotInfoFrame:
    """
    makes frame object for all labels and buttons
    """
    def __init__(self, master):
        self.master = master

        self.guiFrame = tk.Frame(self.master)
        self.guiFrame.grid(row=1, column=2, sticky='news')


        ##### labels #####

        #orientation label
        self.orientationLabelText = tk.StringVar()
        self.orientationLabel = tk.Label(self.guiFrame, textvariable=self.orientationLabelText)

        #distance moved label
        self.distanceMovedLabelText = tk.StringVar()
        self.distanceMovedLabel = tk.Label(self.guiFrame, textvariable=self.distanceMovedLabelText)

        #current command label
        self.commandLabelText = tk.StringVar()
        self.commandLabel = tk.Label(self.guiFrame, textvariable=self.commandLabelText)

        #white space
        self.whiteSpaceLabel = tk.Label(self.guiFrame)

        #trailing white space
        self.trailingWhiteSpaceLabel = tk.Label(self.guiFrame)



    def placeObjects(self):
        """
        places all labels and sets default value
        """

        self.orientationLabel.grid(row=0, column=8, columnspan=1, sticky='w')
        self.distanceMovedLabel.grid(row=1, column=8, columnspan=1, sticky='w')
        self.commandLabel.grid(row=2, column=8, columnspan=1, sticky='w')

        self.whiteSpaceLabel.grid(row=0, column=0, columnspan=7, rowspan=3, sticky='news')
        self.trailingWhiteSpaceLabel.grid(row=0, column=4, sticky='news')


        #configures trailing white space to be eaten
        self.guiFrame.grid_columnconfigure(4, weight=1)


        self.orientationLabelText.set("orientation: ")
        self.distanceMovedLabelText.set("distance moved: N/A                ")
        self.commandLabelText.set("command: ")
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import time
import math
import stopwatch


class robot:
    """
    contains all robot move functions
    """
    def __init__(self, fieldSize=None, tkobj=None, canvas=None, diameterOfWheel=4.1, controlPanelFrame=None, robotInfoFrame=None):
        self.controlPanelFrame = controlPanelFrame
        self.robotInfoFrame = robotInfoFrame

        self.canvas = canvas
        self.master = tkobj

        self.diameterOfWheel = diameterOfWheel
        self.fieldSize = fieldSize
        self.reversed = 0

        self.sqaure = None
        self.line = None

        self.squareVertexes = []
        self.lineVertexes = []


    def __calcSleepTime(self, distance, iterations):
        return (distance / (10*(2 ** self.controlPanelFrame.speed))) / iterations



    def __calcCenters(self):
        """
        returns center of each polygon
        """
        xVals = []
        yVals = []

        for i in range(0, len(self.squareVertexes)): #calculates square center
            xVals.append(self.squareVertexes[i][0])
            yVals.append(self.squareVertexes[i][1])

        xVals.sort(reverse=True)
        yVals.sort(reverse=True)

        greatestX = xVals[0]
        greatestY = yVals[0]

        xVals.sort()
        yVals.sort()

        leastX = xVals[0]
        leastY = yVals[0]

        x = abs(((greatestX - leastX)/2)) + leastX
        y = abs(((greatestY - leastY)/2)) + leastY

        center = [x, y]


        return center




    def __rotate(self, Angle, pivotPoint):
        """
        rotates the robot simulating one side of the
        chassis moving
        """
        center = (pivotPoint[0], pivotPoint[1])
        angle = math.radians(Angle)
        cos_val = math.cos(angle)
        sin_val = math.sin(angle)
        cx, cy = center

        new_points = []
        for x_old, y_old in self.squareVertexes:
            x_old -= cx
            y_old -= cy
            x_new = x_old * cos_val - y_old * sin_val
            y_new = x_old * sin_val + y_old * cos_val
            new_points.append([x_new + cx, y_new + cy])


        angle = math.radians(Angle)
        cos_val = math.cos(angle)
        sin_val = math.sin(angle)
        cx, cy = center

        new_points2 = []
        for x_old, y_old in self.lineVertexes:
            x_old -= cx
            y_old -= cy
            x_new = x_old * cos_val - y_old * sin_val
            y_new = x_old * sin_val + y_old * cos_val
            new_points2.append([x_new + cx, y_new + cy])

        self.squareVertexes = new_points
        self.lineVertexes = new_points2



    def __rotateInPlace(self, Angle):
        """
        rotates the robot in place simulating both sides of the chassis
        moving
        """
```

```
114:        center = self.__calcCenters()
115:
116:
117:        center = (center[0], center[1])
118:
119:        angle = math.radians(Angle) #moves square
120:        cos_val = math.cos(angle)
121:        sin_val = math.sin(angle)
122:        cx, cy = center
123:        new_points = []
124:        for x_old, y_old in self.squareVertexes:
125:            x_old -= cx
126:            y_old -= cy
127:            x_new = x_old * cos_val - y_old * sin_val
128:            y_new = x_old * sin_val + y_old * cos_val
129:            new_points.append([x_new + cx, y_new + cy])
130:
131:        angle = math.radians(Angle) #moves line
132:        cos_val = math.cos(angle)
133:        sin_val = math.sin(angle)
134:        cx, cy = center
135:        new_points2 = []
136:        for x_old, y_old in self.lineVertexes:
137:            x_old -= cx
138:            y_old -= cy
139:            x_new = x_old * cos_val - y_old * sin_val
140:            y_new = x_old * sin_val + y_old * cos_val
141:            new_points2.append([x_new + cx, y_new + cy])
142:
143:
144:        self.squareVertexes = new_points
145:        self.lineVertexes = new_points2
146:
147:
148:
149:    def __move(self, units):
150:        """
151:        simulates the robot moving in a straight line
152:        """
153:        quadrants = {  #quadrant: [xVal, yVal]
154:            1:[1, -1],
155:            2:[-1, -1],
156:            3:[-1, 1],
157:            4:[1, 1]
158:            }
159:
160:
161:        #determine quadrant of final position
162:        #used to determine if robot needs to add or subtract
163:        #x and y value to move in that direction
164:
165:        if units > 0 and (self.orientationDegrees >= 0 and self.orientationDegrees < 90):
166:            quadrant = 1
167:        elif units < 0 and (self.orientationDegrees >= 180 and self.orientationDegrees <= 270):
168:            quadrant = 1
169:
170:        elif units > 0 and (self.orientationDegrees >= 90 and self.orientationDegrees <= 180):
171:            quadrant = 2
172:        elif units < 0 and (self.orientationDegrees > 270 and self.orientationDegrees < 360):
173:            quadrant = 2
174:
175:        elif units > 0 and (self.orientationDegrees >= 180 and self.orientationDegrees < 270):
176:            quadrant = 3
177:        elif units < 0 and (self.orientationDegrees >= 0 and self.orientationDegrees <= 90):
178:            quadrant = 3
179:
180:        elif units > 0 and (self.orientationDegrees >= 270 and self.orientationDegrees < 360):
181:            quadrant = 4
182:        elif units < 0 and (self.orientationDegrees > 90 and self.orientationDegrees < 180):
183:            quadrant = 4
184:
185:
186:
187:        vals = quadrants.get(quadrant)
188:        xPol = vals[0]
189:        yPol = vals[1]
190:
191:        #absolute value simulates reference angle
192:        #trig functions of reference angles are positive
193:        x = xPol * abs(math.cos(math.radians(self.orientationDegrees)))
194:        y = yPol * abs(math.sin(math.radians(self.orientationDegrees)))
195:
196:        d = (math.sqrt((x**2) + (y**2)))
197:        distanceMoved = 0
198:
199:        stp = stopwatch.stopwatch()
200:        stp.start()
201:
202:        iterations = int(abs(units)/d)
203:        for i in range(0, iterations): #move animation
204:            if not self.controlPanelFrame.keepRunning: #allows for pause
205:                while not self.controlPanelFrame.keepRunning:
206:                    time.sleep(0.1)
207:                    self.master.update()
208:
209:            self.canvas.move(self.square, x, y)
210:            self.canvas.move(self.line, x, y)
211:
212:            self.master.update()
213:            time.sleep(self.__calcSleepTime(self.__encoderTicks(abs(units)), iterations))
214:
215:            eus = self.__encoderTicks(d) #shows distance moved
216:            distanceMoved = distanceMoved + eus
217:
218:            self.__updateDistanceLabel(str(round(distanceMoved, 2)) ,"encoder ticks")
219:
220:
221:        #updates vertices of square and line
222:        xChange = xPol * abs(math.cos(math.radians(self.orientationDegrees)) * abs(units))
223:        yChange = yPol * abs(math.sin(math.radians(self.orientationDegrees)) * abs(units))
224:
225:        self.squareVertexes = [
226:            [self.squareVertexes[0][0] + xChange, self.squareVertexes[0][1] + yChange],
```

```
            [self.squareVertexes[1][0] + xChange, self.squareVertexes[1][1] + yChange],
            [self.squareVertexes[2][0] + xChange, self.squareVertexes[2][1] + yChange],
            [self.squareVertexes[3][0] + xChange, self.squareVertexes[3][1] + yChange]
            ]

        self.lineVertexes = [
            [self.lineVertexes[0][0] + xChange, self.lineVertexes[0][1] + yChange],
            [self.lineVertexes[1][0] + xChange, self.lineVertexes[1][1] + yChange],
            [self.lineVertexes[2][0] + xChange, self.lineVertexes[2][1] + yChange],
            [self.lineVertexes[3][0] + xChange, self.lineVertexes[3][1] + yChange]
            ]



    def __update(self):
        """
        updates tkinter canvas so robot can be seen moving
        and also allows for a pause
        """
        self.canvas.delete(self.square)
        self.canvas.delete(self.line)

        self.square = self.canvas.create_polygon(self.squareVertexes,
                                         outline="black",
                                         fill="#949596",
                                         width=3)

        self.line = self.canvas.create_polygon(self.lineVertexes, width=3)


        if not self.controlPanelFrame.keepRunning: #allows for pause during turns
            while not self.controlPanelFrame.keepRunning:
                time.sleep(0.1)
                self.master.update()

        self.master.update()


    def __updateDistanceLabel(self, distance=0, units=""):
        """
        updates distance travelled label
        """
        text = "distance moved: " + str(distance) + " " + units
        while len(text) < 38:
            text = text + " "
        self.robotInfoFrame.distanceMovedLabelText.set(text)

        self.master.update()


    def __pixels(self, rotationUnits):
        """
        converts encoder ticks to pixels
        """
        revolutions = rotationUnits / 360
        inches = revolutions * (self.diameterOfWheel * math.pi)
        pixelsToMove = (inches * self.fieldSize) / 144
        print(rotationUnits, pixelsToMove)
        return pixelsToMove


    def __encoderTicks(self, pixels):
        """
        converts pixels to encoder ticks
        """
        inches = pixels * (144 / self.fieldSize)
        revolutions = inches / (self.diameterOfWheel * math.pi)
        encoderTicks = revolutions * 360

        return encoderTicks


    def inches(self, pixels):
        """
        converts pixels to inches
        """
        inches = pixels * (144 / self.fieldSize)

        return inches


    def show(self, angle=0, position=[[0, 0], [0, 0]]):
        """
        starting function that shows the robot based on two coordinates
        if more than two coordinates are given use different show function
        """

        #if len(position) > 2:
        if 1:
            x1 = position[0][0]
            y1 = position[0][1]
            x2 = position[1][0]
            y2 = position[1][1]

            self.orientationDegrees = 90
            self.squareVertexes = [
                [x1, y1],
                [x2, y1],
                [x2, y2],
                [x1, y2]
                ]

            self.sizeOfSquare = abs(self.squareVertexes[0][0] - self.squareVertexes[1][0])

            y2 = (y2 - y1) / 4
            self.lineVertexes = [
                [x1, y1+y2],
                [x2, y1+y2],
                [x2, y1+y2+4],
                [x1, y1+y2+4]
                ]

#       else:
```

```
340:  #         x1 = position[0][0]
341:  #         x2 = position[1][0]
342:  #         x3 = position[2][0]
343:  #         x4 = position[3][0]
344:  #
345:  #         y1 = position[0][1]
346:  #         y2 = position[1][1]
347:  #         y3 = position[2][1]
348:  #         y4 = position[3][1]
349:  #
350:  #         self.squareVertexes = position
351:  #         self.lineVertexes = []
352:
353:
354:          self.square = self.canvas.create_polygon(self.squareVertexes,
355:                                                   outline="black",
356:                                                   fill="#949596",
357:                                                   width=3)
358:
359:          turn = 90 - (angle % 360)
360:
361:          self.line = self.canvas.create_polygon(self.lineVertexes, width=3)
362:          self.__rotateInPlace(turn)
363:          self.__update()
364:
365:          self.orientationDegrees = angle % 360
366:
367:
368:
369:
370:      def reverse(self):
371:          """
372:          reverses orientation of the robot
373:          """
374:          self.reversed = not(self.reversed)
375:
376:
377:
378:
379:      def forward(self, rotationUnits):
380:          """
381:          moves the robot forward and straight
382:          """
383:          self.robotInfoFrame.commandLabelText.set(("forward " + str(rotationUnits)))
384:
385:          if self.reversed:
386:              rotationUnits = 0 - rotationUnits
387:
388:          pixelsToMove = self.__pixels((rotationUnits))
389:
390:          self.__move(pixelsToMove)
391:          self.__update()
392:
393:
394:
395:
396:      def backward(self, rotationUnits):
397:          """
398:          moves the robot backwards and straight
399:          """
400:          self.robotInfoFrame.commandLabelText.set(("backward " + str(rotationUnits)))
401:
402:          rotationUnits = 0 - rotationUnits
403:          if self.reversed:
404:              rotationUnits = 0 - rotationUnits
405:
406:          pixelsToMove = self.__pixels(rotationUnits)
407:
408:
409:          self.__move(pixelsToMove)
410:          self.__update()
411:
412:
413:      def leftSide(self, angle):
414:          """
415:          turns the robot right so that only one side is moving
416:          """
417:          self.robotInfoFrame.commandLabelText.set(("leftSide " + str(angle)))
418:
419:          if self.reversed:
420:              angle = 0 - angle
421:
422:          pivotPoints = self.squareVertexes[2]
423:
424:          turned = 0
425:          orientation = angle / abs(angle) #to account for negative turns
426:          toMove = orientation * .5
427:          while turned < abs(angle): #turn to specified angle
428:              self.__rotate(toMove, pivotPoints)
429:              self.__update()
430:              time.sleep(self.__calcSleepTime(angle, angle))
431:
432:              turned += .5
433:              self.__updateDistanceLabel(str(round(turned, 2)), "degrees")
434:
435:              self.orientationDegrees = (self.orientationDegrees - toMove) % 360
436:              self.robotInfoFrame.orientationLabelText.set("orientation: " + str(self.orientationDegrees))
437:
438:
439:
440:      def rightSide(self, angle):
441:          """
442:          turns the robot left so that only the right side is moving
443:          """
444:          self.robotInfoFrame.commandLabelText.set(("rightSide " + str(angle)))
445:
446:          angle = 0 - angle
447:          if self.reversed:
448:              angle = 0 - angle
449:
450:          pivotPoints = self.squareVertexes[3]
451:
452:          turned = 0
```

```
453:         orientation = angle / abs(angle) #to account for negative turns
454:         toMove = (-1 * orientation * .5)
455:         while turned < abs(angle): #turn to specified angle
456:             self.__rotate(toMove, pivotPoints)
457:             self.__update()
458:             time.sleep(self.__calcSleepTime(angle, angle))
459:             turned += .5
460:             self.__updateDistanceLabel(str(round(turned, 2)), "degrees")
461:
462:             self.orientationDegrees = (self.orientationDegrees + toMove) % 360
463:             self.robotInfoFrame.orientationLabelText.set("orientation: " + str(self.orientationDegrees))
464:
465:
466:
467:     def turnLeft(self, angle):
468:         """
469:         turn in place left
470:         """
471:         self.robotInfoFrame.commandLabelText.set(("turnLeft " + str(angle)))
472:
473:         if self.reversed:
474:             angle = 0 - angle
475:
476:         turned = 0
477:         orientation = angle / abs(angle) #to account for negative turns
478:         toMove = -1 * orientation * .5
479:
480:         while turned < abs(angle): #turn to specified angle
481:             self.__rotateInPlace(toMove)
482:             self.__update()
483:             time.sleep(self.__calcSleepTime(angle, angle))
484:             turned += .5
485:             self.__updateDistanceLabel(str(round(turned, 2)), "degrees")
486:
487:             self.orientationDegrees = (self.orientationDegrees - toMove) % 360
488:             self.robotInfoFrame.orientationLabelText.set("orientation: " + str(self.orientationDegrees))
489:
490:
491:
492:
493:     def turnRight(self, angle):
494:         """
495:         turn in place right
496:         """
497:         self.robotInfoFrame.commandLabelText.set(("turnRight " + str(angle)))
498:
499:         if self.reversed:
500:             angle = 0 - angle
501:
502:         turned = 0
503:         orientation = angle / abs(angle) #to account for negative turns
504:         toMove = .5 * orientation
505:
506:         while turned < abs(angle): #turn to specified angle
507:             self.__rotateInPlace(toMove)
508:             self.__update()
509:             time.sleep(self.__calcSleepTime(angle*2, angle))
510:             turned += .5
511:             self.__updateDistanceLabel(str(round(turned, 2)), "degrees")
512:
513:
514:             self.orientationDegrees = (self.orientationDegrees - toMove) % 360
515:             self.robotInfoFrame.orientationLabelText.set("orientation: " + str(self.orientationDegrees))
516:
517:
518:
519:
520:
521:
```

```
1:  #!/usr/bin/env python3
2:  # -*- coding: utf-8 -*-
3:
4:  import tkinter as tk
5:
6:  class runningFrame:
7:      """
8:      makes frame for running label
9:      """
10:     def __init__(self, master):
11:         self.master = master
12:
13:         self.guiFrame = tk.Frame(self.master)
14:         self.guiFrame.grid(row=2, column=0, sticky='news')
15:
16:
17:         ##### labels #####
18:
19:         #running label
20:         self.runningLabelText = tk.StringVar()
21:         self.runningLabel = tk.Label(self.guiFrame, textvariable=self.runningLabelText, font=("Courier", 15))
22:
23:
24:
25:     def placeObjects(self):
26:         """
27:         places labels
28:         """
29:
30:         self.runningLabel.grid(sticky='w', row=0, column=0)
31:
32:         self.guiFrame.grid_columnconfigure(0, weight=1)
```

```
1:  #!/usr/bin/env python3
2:  # -*- coding: utf-8 -*-
3:
4:  import time
5:
6:  class stopwatch:
7:
8:      def __init__(self):
9:          self.beginning = None
10:
11:     def start(self):
12:         """
13:         starts stopwatch
14:         """
15:         self.beginning = time.time()
16:
17:     def stop(self):
18:         """
19:         stops stopwatch and returns
20:         time elapsed
21:         """
22:         end = time.time()
23:         timeElapsed = end - self.beginning
24:
25:         return timeElapsed
```

1: **Update** *creds.json for google docs spreadsheet that will be modified*
2:
3: in "Scout.py" change sku in class constructor to sku of tournament

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 24 22:08:27 2019

@author: aiden
"""

import requests
import json
import stopwatch
from usefultools import split
import multiprocessing as mp
import gspread
from oauth2client.client import SignedJwtAssertionCredentials
import time


class getData:
    def __init__(self):
        self.sku = ""

        self.sheet_name = ""

        self.matchesWonUrl = 'https://api.vexdb.io/v1/get_matches?round=2'
        self.skillsUrl = 'https://api.vexdb.io/v1/get_skills?'
        self.pointsUrl = 'https://api.vexdb.io/v1/get_rankings?sku=' + self.sku

        self.teams = []
        self.allTeams = []
        self.ranges ={}

        self.matchesWonData = {}
        self.matchesWonOverallData = {}
        self.combinedSkillsScoreData = {}
        self.driverSkillsScoreData = {}
        self.pointsData = {}
        self.rankingData = {}

        self.NUM_PROCESSES = 20
        self.TEAM_NUM_COL = 2

        self.sheet = None


    def __calcWin(self, team, data):
        """
        calculates if a match was won or not by finding what
        color a team was and the score of the match
        it is a win if the teams color score more points
        """
        colors = ['red1', 'red2', 'blue1', 'blue2']
        teamColor = ''
        for color in colors:
            if data.get(color) == team:
                c = color.split('1')[0]
                c = c.split('2')[0]
                teamColor = c
                break

        redScore = data.get('redscore')
        blueScore = data.get('bluescore')

        if redScore != blueScore:
            if redScore > blueScore:
                winner = 'red'
            else:
                winner = 'blue'

            if teamColor == 'red' and winner == 'red':
                return 1
            elif teamColor == 'blue' and winner == 'blue':
                return 1
            else:
                return 0

        elif redScore == blueScore and (redScore != 0 and blueScore != 0):
            return -1

        else:
            return None



    def __getMatchesWon(self, teams, q):
        """
        gets win/loss/tie data for a team
        """
        for team in teams:
            team = team.split('\n')[0]
            url = self.matchesWonUrl + '&team=' + team + '&season=Tower Takeover'

            data = requests.get(url)
            if data.status_code == 200 and team != '':

                winStruct = {'wins':0,
                             'losses':0,
                             'ties':0
                            }

                data = json.loads(data.content.decode('utf-8'))
                data = data.get('result')

                for entry in data:
#                    print(team, entry)
                    x = self.__calcWin(team, entry)
                    if x == 1:
                        winStruct['wins'] = winStruct.get('wins') + 1
                    elif x == 0:
                        winStruct['losses'] = winStruct.get('losses') + 1
                    elif x == -1:
                        winStruct['ties'] = winStruct.get('ties') + 1
```

```
114:                 else:
115:                     pass
116:             q.put({team: winStruct})
117:
118:
119:
120:     def __getDriverSkillsScore(self, teams, q):
121:         """
122:         gets driver skills score for a team
123:         """
124:         for team in teams:
125:             team = team.split('\n')[0]
126:             url = self.skillsUrl + 'team=' + team + '&season=Tower Takeover&type=0'
127:
128:             data = requests.get(url)
129:             if data.status_code == 200:
130:
131:                 data = json.loads(data.content.decode('utf-8'))
132:                 data = data.get('result')
133:                 highest = 0
134:
135:                 for item in data:
136:                     value = item.get('score')
137:                     if value > highest:
138:                         highest = value
139:
140:                 q.put({team: highest})
141:
142:
143:     def __getCombinedSkillsScore(self, teams, q):
144:         """
145:         gets the combined skills score for a team
146:         (driver and programming)
147:         """
148:         for team in teams:
149:             team = team.split('\n')[0]
150:             url2 = self.skillsUrl + 'team=' + team + '&season=Tower Takeover&type=2'
151:
152:             data = requests.get(url2)
153:             if data.status_code == 200:
154:
155:
156:                 data = json.loads(data.content.decode('utf-8'))
157:                 data = data.get('result')
158:                 highest = 0
159:
160:                 for item in data:
161:                     value = item.get('score')
162:                     if value > highest:
163:                         highest = value
164:                 q.put({team: highest})
165:
166:
167:     def __getPoints(self, teams, q):
168:         """
169:         gets win points/auton points/strength points/calculated
170:         contribution to win margin data for a team
171:         """
172:         for team in teams:
173:             team = team.split('\n')[0]
174:             url = self.pointsUrl + '&team=' + team + '&season=Tower Takeover&type=2'
175:
176:             data = requests.get(url)
177:             if data.status_code == 200:
178:                 try:
179:                     data = json.loads(data.content.decode('utf-8'))
180:                     data = data.get('result')[0]
181:                     pointsStruct = {
182:                         'wp':data.get('wp'),
183:                         'ap':data.get('ap'),
184:                         'sp':data.get('sp'),
185:                         'ccwm':data.get('ccwm')
186:                         }
187:                     q.put({team: pointsStruct})
188:
189:                 except IndexError:
190:                     q.put({team: 'N/A'})
191:
192:     def __getRanking(self, teams, q):
193:         """
194:         gets the ranking of a team at an event
195:         """
196:         for team in teams:
197:             team = team.split('\n')[0]
198:             url = self.pointsUrl + '&team=' + team + '&season=Tower Takeover&type=2'
199:             #print(url)
200:
201:             data = requests.get(url)
202:             if data.status_code == 200:
203:
204:                 data = json.loads(data.content.decode('utf-8'))
205:                 try:
206:                     data = data.get('result')[0]
207:                     q.put({team: data.get('rank')})
208:
209:                 except IndexError:
210:                     q.put({team: 'N/A'})
211:
212:
213:     def __parralellise(self, func, returnDict):
214:         """
215:         processes the data in parrallel so that if there is a lot of
216:         teams the operation can be performed quickly
217:         """
218:         queues = []
219:         processes = []
220:         for i in range(len(self.teams)):
221:             queues.append(mp.Queue())
222:             p = mp.Process(target=func, args=(self.teams[i], queues[i],))
223:             processes.append(p)
224:             p.daemon = True
225:             p.start()
226:         for process in processes:
```

```
227:            process.join()
228:
229:        for q in queues:
230:            while not q.empty():
231:                returnDict.update(q.get(timeout=.1))
232:
233:
234:
235:
236:
237:    def openSheet(self):
238:        """
239:        opens the sheet
240:        need to change workbook to the sheet that will be edited
241:        and the sheet name to the sheet that corresponds with the
242:        data that will be collected
243:        """
244:        json_key = json.load(open('/home/aiden/Documents/google_credentials/creds.json'))
245:        scope = ['https://spreadsheets.google.com/feeds',
246:        'https://www.googleapis.com/auth/drive']
247:
248:        credentials = SignedJwtAssertionCredentials(json_key['client_email'], json_key['private_key'].encode(), scope)
249:
250:        file = gspread.authorize(credentials)
251:        workbook = file.open("536C_Scouting")
252:        self.sheet = workbook.worksheet(self.sheet_name)
253:
254:
255:
256:
257:
258:    def getTeams(self):
259:        """
260:        gets the teams by reading the data in the sheet
261:        this looks for data to find by comparing the first
262:        row to a list of valid headers then removes the headers
263:        from the list of cells to be updated
264:        """
265:        valid_headers = [
266:                'Rank',
267:                'WP/AP/SP/CCWM',
268:                'W-L-T (today)',
269:                'W-L-T (overall season)',
270:                'driver skills',
271:                'combined skills score'
272:                ]
273:
274:        self.allTeams = self.sheet.col_values(1) #column that teams are stored in
275:        self.allTeams.pop(0) #remove header
276:        headers = self.sheet.row_values(1) #remove header
277:        row = 2
278:        column = 1
279:        for header in headers:
280:            if header in valid_headers:
281:                start = chr(ord('@')+column) + str(row)
282:                end = chr(ord('@')+column) + str((len(self.allTeams) + 1))
283:                string = start + ':' + end
284:                range_= self.sheet.range(string)
285:                self.ranges.update({header:range_})
286:            column += 1
287:
288:        #limits num processes to the number of teams
289:        if self.NUM_PROCESSES > len(self.allTeams):
290:            self.NUM_PROCESSES = len(self.allTeams)
291:
292:        self.teams = list(split.split(self.allTeams, self.NUM_PROCESSES))
293:
294:
295:
296:
297:
298:    def collect(self):
299:        """
300:        collects all the data if that data is a valid header
301:        """
302:        if 'W-L-T (overall season)' in self.ranges.keys():
303:            self.__parralellise(self.__getMatchesWon, self.matchesWonOverallData)
304:
305:        self.matchesWonUrl = self.matchesWonUrl + '&sku=' + self.sku
306:
307:        if 'W-L-T (today)' in self.ranges.keys():
308:            self.__parralellise(self.__getMatchesWon, self.matchesWonData)
309:
310:        if 'combined skills score' in self.ranges.keys():
311:            self.__parralellise(self.__getCombinedSkillsScore, self.combinedSkillsScoreData)
312:        if 'driver skills' in self.ranges.keys():
313:            self.__parralellise(self.__getDriverSkillsScore, self.driverSkillsScoreData)
314:        if 'WP/AP/SP/CCWM' in self.ranges.keys():
315:            self.__parralellise(self.__getPoints, self.pointsData)
316:        if 'Rank' in self.ranges.keys():
317:            self.__parralellise(self.__getRanking, self.rankingData)
318:
319:
320:
321:    def printData(self):
322:        """
323:        prints the data for each team to be used for debugging
324:        purposes
325:        """
326:        if self.matchesWonData:
327:            #print("w-l-t")
328:            for team in self.matchesWonData.keys():
329:                try:
330:                    wins = self.matchesWonData.get(team)
331:                    record = [wins.get('wins'), wins.get('losses'), wins.get('ties')]
332:                    formattedRecord = str(record[0]) + '-' + str(record[1]) + '-' + str(record[2])
333:                    print(formattedRecord)
334:                except AttributeError:
335:                    print('N/A')
336:            for i in range(20):
337:                print("")
338:
339:        if self.combinedSkillsScoreData:
```

```
340:            print("combined skills")
341:            for team in self.combinedSkillsScoreData.keys():
342:                print(self.combinedSkillsScoreData.get(team))
343:
344:
345:            for i in range(20):
346:                print("")
347:
348:        if self.driverSkillsScoreData:
349:            print("driver skills")
350:            for team in self.driverSkillsScoreData.keys():
351:                print(self.driverSkillsScoreData.get(team))
352:
353:            for i in range(20):
354:                print("")
355:
356:
357:        if self.rankingData:
358:            print("ranking")
359:            for team in self.rankingData.keys():
360:                print(self.rankingData.get(team))
361:
362:            for i in range(20):
363:                print("")
364:
365:
366:        if self.pointsData:
367:            print("point values")
368:            for team in self.pointsData.keys():
369:                try:
370:                    val = self.pointsData.get(team)
371:                    points = (str(val.get('wp'))  + ' / '
372:                            + str(val.get('ap')) + ' / '
373:                            + str(val.get('sp')) + ' / '
374:                            + str(val.get('ccwm'))
375:                            )
376:                    print(points)
377:                except AttributeError:
378:                    print('N/A')
379:            for i in range(6):
380:                print("")
381:
382:
383:
384:
385:
386:    def writeData(self):
387:        """
388:        writes the data in one chunk because the google api
389:        only allows so many edits
390:        by making it only one edit once all the data is collected
391:        this constraing can be worked around
392:        """
393:        cell_list = self.ranges.get('Rank') #write rank
394:        row = 0
395:        for cell in cell_list:
396:            try:
397:                team = self.allTeams[row].split('\n')[0]
398:                data = self.rankingData.get(team)
399:            except:
400:                data = "N/A"
401:            cell.value = data
402:            row += 1
403:
404:        self.sheet.update_cells(cell_list)
405:
406:
407:        cell_list = self.ranges.get('WP/AP/SP/CCWM') #write points
408:        row = 0
409:        for cell in cell_list:
410:            try:
411:                team = self.allTeams[row].split('\n')[0]
412:                val = self.pointsData.get(team)
413:                data = (str(val.get('wp'))  + ' / '
414:                        + str(val.get('ap')) + ' / '
415:                        + str(val.get('sp')) + ' / '
416:                        + str(val.get('ccwm'))
417:                        )
418:                cell.value = data
419:            except AttributeError:
420:                cell.value = 'N/A'
421:            except IndexError:
422:                cell.value = 'N/A'
423:            row += 1
424:        self.sheet.update_cells(cell_list)
425:
426:
427:
428:        cell_list = self.ranges.get('W-L-T (today)') #write w-l-t
429:        row = 0
430:        for cell in cell_list:
431:            try:
432:                team = self.allTeams[row].split('\n')[0]
433:                wins = self.matchesWonData.get(team)
434:                record = [wins.get('wins'), wins.get('losses'), wins.get('ties')]
435:                data = str(record[0]) + '-' + str(record[1]) + '-' + str(record[2])
436:
437:                cell.value = data
438:            except AttributeError:
439:                cell.value = 'N/A'
440:            row += 1
441:
442:        self.sheet.update_cells(cell_list)
443:
444:
445:
446:        cell_list = self.ranges.get('W-L-T (overall season)') #write w-l-t for
447:        row = 0                                 #overall season
448:        for cell in cell_list:
449:            try:
450:                team = self.allTeams[row].split('\n')[0]
451:                wins = self.matchesWonOverallData.get(team)
452:                record = [wins.get('wins'), wins.get('losses'), wins.get('ties')]
```

```
453:                data = str(record[0]) + '-' + str(record[1]) + '-' + str(record[2])
454:
455:                cell.value = data
456:            except AttributeError:
457:                cell.value = 'N/A'
458:            row += 1
459:
460:        self.sheet.update_cells(cell_list)
461:
462:
463:
464:
465:        cell_list = self.ranges.get('driver skills')#write driver skills
466:        row = 0
467:        for cell in cell_list:
468:            team = self.allTeams[row].split('\n')[0]
469:            data = self.driverSkillsScoreData.get(team)
470:            cell.value = data
471:
472:            row += 1
473:
474:        self.sheet.update_cells(cell_list)
475:
476:
477:
478:        cell_list = self.ranges.get('combined skills score') #write combined skills
479:        row = 0
480:        for cell in cell_list:
481:            team = self.allTeams[row].split('\n')[0]
482:            data = self.combinedSkillsScoreData.get(team)
483:            cell.value = data
484:
485:            row += 1
486:
487:        self.sheet.update_cells(cell_list)
488:
489:
490:
491:
492:
493:
494:
495:
496:  while 1: #collect data every so often
497:      stp = stopwatch.stopwatch()
498:      stp.start()
499:      d = getData()
500:      d.openSheet()
501:
502:      d.getTeams()
503:      d.collect()
504:      #print(d.rankingData)
505:      d.writeData()
506:      print("time taken:", stp.stop(), "sec")
507:
508:      time.sleep(45)
509:
510:
511:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 28 11:36:52 2019

@author: aiden
"""
import requests
import json
import stopwatch
from usefultools import split
import multiprocessing as mp
import gspread
from oauth2client.client import SignedJwtAssertionCredentials
import time


class getTeams:
    """
    class for getting teams at a tournament
    based on the tournaments sku
    """
    def __init__(self):
        self.sheet_name = "Signature"

        #xavier tournament
        #sku = 'RE-VRC-19-8551'
        #menasha tournament
        #sku = "RE-VRC-19-8167"
        #smc tournament
        #sku = "RE-VRC-19-0539"
        #league
        #sku = "RE-VRC-19-9815&"
        #Signature event
        sku = "RE-VRC-19-0203"

        self.url = "https://api.vexdb.io/v1/get_teams?round=5?&sku=" + sku
        self.elimsUrl = "https://api.vexdb.io/v1/get_teams?round=5?&sku=" + sku + '&matchnum='

        #legacy version that only works if matches have started
        #self.url = 'https://api.vexdb.io/v1/get_matches?round=2?&sku=' + sku
        #self.elimsUrl = 'https://api.vexdb.io/v1/get_matches?&sku=' + sku + '&matchnum='

        self.allTeams = []
        self.elimTeams = []

        self.COLLECT_ELIMS = 0
        self.COLLECT_TEAMS = 1

        self.NUM_PROCESSES = 50

        self.sheet = None

    def __getAllTeams(self, dicts, queue):
        """
        gets all teams that are registered
        """
        for entry in dicts:
            print(entry.get("number"))
            queue.put(entry.get("number"))


    def __getElimTeams(self, dicts, queue):
        """
        gets teams that are in the elimination matches
        does not work need to update
        """
        print(dicts)
        for entry in dicts:
            print(entry.get("number"))
            queue.put(entry.get("number"))

    def __parralellise(self, func, returnList, dicts):
        """
        starts threads that look through lists of entries
        for teams that are at the tournament
        this is used so that at events like worlds it does
        not take forever to run
        """
        queues = []
        processes = []
        for i in range(len(dicts)):
            queues.append(mp.Queue())
            p = mp.Process(target=func, args=(dicts[i], queues[i],))
            processes.append(p)
            p.daemon = True
            p.start()
        for process in processes:
            process.join()

        for q in queues:
            while not q.empty():
                returnList.append(q.get(timeout=.1))


    def collect(self):
        """
        collects the data from vexdb and splits it into
        entries based on self.NUM_PROCESSES so that data
        can be parsed faster especially for events like worlds
        """
        data = requests.get(self.url)
        if data.status_code == 200 and self.COLLECT_TEAMS:
            data = json.loads(data.content.decode('utf-8'))
            data = data.get('result')
            data = list(split.split(data, self.NUM_PROCESSES))

            self.__parralellise(self.__getAllTeams, self.allTeams, data)

            self.allTeams = list(set(self.allTeams))

```

```
114:
115:        if self.COLLECT_ELIMS:
116:            links = []
117:            for num in range(1, 9):
118:                match = 'R16 #' + str(num) + '-1'
119:                link = self.elimsUrl + match
120:                links.append(link)
121:
122:            data = []
123:
124:            for url in links:
125:                response = requests.get(url)
126:                if response.status_code == 200:
127:                    response = json.loads(response.content.decode('utf-8'))
128:                    allData = response.get('result') + data #merge lists
129:
130:
131:    #       data = requests.get(self.elimsUrl)
132:    #       if data.status_code == 200 and self.COLLECT_ELIMS:
133:    #           data = json.loads(data.content.decode('utf-8'))
134:    #           data = data.get('result')
135:            data = list(split.split(allData, self.NUM_PROCESSES))
136:
137:            self.__parralellise(self.__getElimTeams, self.elimTeams, data)
138:
139:            self.elimTeams = list(set(self.elimTeams))
140:
141:
142:
143:
144:
145:
146:
147:
148:    def printTeams(self):
149:        """
150:        prints the teams out
151:        used for debugging
152:        """
153:        for team in self.allTeams:
154:            print(team)
155:
156:        print("")
157:        print("")
158:
159:        for team in self.elimTeams:
160:            print(team)
161:
162:
163:
164:    def openSheet(self):
165:        """
166:        opens the sheet and loads the work book
167:        need to change the workbook to the file name
168:        and the sheet to the sheet that will be edited
169:        """
170:        json_key = json.load(open('/home/aiden/Documents/google_credentials/creds.json'))
171:        scope = scope = ['https://spreadsheets.google.com/feeds',
172:                         'https://www.googleapis.com/auth/drive']
173:
174:        credentials = SignedJwtAssertionCredentials(json_key['client_email'], json_key['private_key'].encode(), scope)
175:
176:        file = gspread.authorize(credentials)
177:        workbook = file.open("536C_Scouting")
178:        self.sheet = workbook.worksheet(self.sheet_name)
179:
180:
181:    def writeData(self, column):
182:        """
183:        writes the data in one chunk because the api only allows
184:        so many operations
185:        """
186:        if self.COLLECT_TEAMS:
187:            #leave room for header by setting to two and adding 1
188:            start = chr(ord('@')+column) + '2'
189:            end = chr(ord('@')+column) + str(len(self.allTeams) + 1)
190:            rang = start + ':' + end
191:            print(rang)
192:            cell_list = self.sheet.range(rang)
193:
194:            x = 0
195:            for cell in cell_list:
196:                cell.value = self.allTeams[x]
197:                x += 1
198:
199:            self.sheet.update_cells(cell_list)
200:
201:
202:        if self.COLLECT_ELIMS:
203:            start = chr(ord('@')+column) + '2'
204:            end = chr(ord('@')+column) + str(len(self.elimTeams) + 1)
205:            rang = start + ':' + end
206:            print(rang)
207:            cell_list = self.sheet.range(rang)
208:
209:            x = 0
210:            for cell in cell_list:
211:                cell.value = self.elimTeams[x]
212:                x += 1
213:
214:            self.sheet.update_cells(cell_list)
215:
216:
217:
218:
219:    g = getTeams()
220:    g.collect()
221:    g.printTeams()
222:    g.openSheet()
223:    print("sheet opened")
224:    g.writeData(1)
```

```
1:  #!/usr/bin/env python3
2:  # -*- coding: utf-8 -*-
3:
4:  import time
5:
6:  class stopwatch:
7:
8:      def __init__(self):
9:          self.beginning = None
10:
11:     def start(self):
12:         """
13:         starts stopwatch
14:         """
15:         self.beginning = time.time()
16:
17:     def stop(self):
18:         """
19:         stops stopwatch and returns
20:         time elapsed
21:         """
22:         end = time.time()
23:         timeElapsed = end - self.beginning
24:
25:         return timeElapsed
```

```
1: #!/bin/bash
2: prosv5 make
3: prosv5 upload --slot 2
4: prosv5 v5 run 2
5: prosv5 terminal
```

```
{
   "internal_motor_pid":[1, 0, 0, 0],
   "tilter_pid_consts":[1, 0, 0, 0],

   "front_right_port":20,
   "back_left_port":4,
   "front_left_port":18,
   "back_right_port":2,
   "left_intake_port":6,
   "right_intake_port":8,
   "tilter_port":19,
   "lift_port":9,

   "front_right_reversed":0,
   "back_left_reversed":1,
   "front_left_reversed":1,
   "back_right_reversed":0,
   "left_intake_reversed":1,
   "right_intake_reversed":0,
   "tilter_reversed":1,
   "lift_reversed":0,

   "tilter_setpoints":[100, 300, 400, 500],
   "lift_setpoints":[100, 300, 400, 500],
   "intake_speeds":[-63, -30, 0, 30, 63],

   "autons":{
      "auton1":"location_of_auton1.json",
      "auton2":"location_of_auton2.json",
      "auton3":"location_of_auton3.json",
      "auton4":"location_of_auton4.json"
   }
}
```

```
1: #!/bin/bash
2: echo "paste stack locations"
3: while true;
4: do
5:     read -p "" stack
6:     arm-none-eabi-addr2line --demangle --inlines -faps -e bin/monolith.elf $stack
7: done
8:
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Oct  2 18:40:57 2019

@author: aiden

graphs items in queue vs time from a txt file
"""
import matplotlib.pyplot as plt


with open("log.txt") as f:
    data = f.readlines()

x_axis = [] #time (ms)
y_axis = [] #items
for line in data:
    time = line.split(" ")[0]
    items = line.split(" ")[1]
    x_axis.append(time)
    y_axis.append(items)
print(len(x_axis))
for i in range(0,9):
    del x_axis[::2]
    del y_axis[::2]
print(len(x_axis))

plt.gcf().set_size_inches(30, 20)
plt.xticks(rotation=45)

plt.plot(x_axis, y_axis)
plt.xlabel('Time (ms)')
plt.ylabel('Items in Queue')

plt.title('Items in Queue vs Time')
plt.savefig("items_vs_time.png", dpi=100, forward=True, bbox_inches='tight', colLoc='center', loc='bottom')

plt.show()


```

```
1:  /**
2:   * @file: ./RobotCode/include/fonts/fonts.h
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/16/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: ../src/objects/lcdCode/fonts/
8:   *
9:   * contains definitions for fonts to use for lvgl
10:  * this is used to make gui more interesting and provide more contrast and have
11:  * the ability to fit more data because of a smaller font
12:  */
13:
14:  #ifndef __FONTS_H__
15:  #define __FONTS_H__
16:
17:
18:  #ifdef USE_DEJAVU_9
19:  LV_FONT_DECLARE(dejavu_9);
20:  extern lv_font_t dejavu_9;
21:  #endif
22:
23:  #ifdef USE_DEJAVU_12
24:  LV_FONT_DECLARE(dejavu_12);
25:  extern lv_font_t dejavu_12;
26:  #endif
27:
28:  #ifdef USE_DEJAVU_16
29:  LV_FONT_DECLARE(dejavu_16);
30:  extern lv_font_t dejavu_16;
31:  #endif
32:
33:  #endif
```

```
#include <csignal>
#include <cstdlib>
#include <errno.h>
#include <fstream>
#include <string>
#include <iostream>
#include <cmath>
#include <cerrno>
#include <cstring>
#include <clocale>

#include "main.h"

#include "Autons.hpp"
#include "DriverControl.hpp"
#include "Configuration.hpp"
#include "objects/controller/controller.hpp"
#include "objects/lcdCode/DriverControl/LCDTask.hpp"
#include "objects/lcdCode/gui.hpp"
#include "objects/lcdCode/TemporaryScreen.hpp"
#include "objects/motors/Motors.hpp"
#include "objects/motors/MotorThread.hpp"
#include "objects/robotChassis/chassis.hpp"
#include "objects/writer/Writer.hpp"


int final_auton_choice;

/**
 * Runs initialization code. This occurs as soon as the program is started.
 *
 * All other competition modes are blocked by initialize; it is recommended
 * to keep execution time for this mode under a few seconds.
 */
 void initialize()
 {
    pros::delay(100); //wait for terminal to start and lvgl
    Configuration* config = Configuration::get_instance();
    config->init();
    config->print_config_options();

    Motors *motors = Motors::get_instance();

    MotorThread* motor_thread = MotorThread::get_instance();
    motor_thread->start_thread();

    final_auton_choice = chooseAuton();

        DriverControlLCD::auton = final_auton_choice;

    std::cout << OptionsScreen::cnfg.use_hardcoded << '\n';
    std::cout << OptionsScreen::cnfg.gyro_turn << '\n';
    std::cout << OptionsScreen::cnfg.accelleration_ctrl << '\n';
    std::cout << OptionsScreen::cnfg.check_motor_tmp << '\n';
    std::cout << OptionsScreen::cnfg.use_previous_macros << '\n';
    std::cout << OptionsScreen::cnfg.record << '\n';

    std::cout << "initalize finished" << "\n";
    lv_scr_load(tempScreen::temp_screen);
 }



/**
 * Runs while the robot is in the disabled state of Field Management System or
 * the VEX Competition Switch, following either autonomous or opcontrol. When
 * the robot is enabled, this task will exit.
 */
void disabled() {}


/**
 * Runs after initialize(), and before autonomous when connected to the Field
 * Management System or the VEX Competition Switch. This is intended for
 * competition-specific initialization routines, such as an autonomous selector
 * on the LCD.
 *
 * This task will exit when the robot is enabled and autonomous or opcontrol
 * starts.
 */
void competition_initialize() {}


/**
 * Runs the user autonomous code. This function will be started in its own task
 * with the default priority and stack size whenever the robot is enabled via
 * the Field Management System or the VEX Competition Switch in the autonomous
 * mode. Alternatively, this function may be called in initialize or opcontrol
 * for non-competition testing purposes.
 *
 * If the robot is disabled or communications is lost, the autonomous task
 * will be stopped. Re-enabling the robot will restart the task, not re-start it
 * from where it left off.
 */
void autonomous() {
    lv_scr_load(tempScreen::temp_screen);
    Autons auton;
    switch(final_auton_choice)
    {
        case 1:
          break;

        case 2:
          auton.auton1(OptionsScreen::cnfg);
          break;

        case 3:
          auton.auton2(OptionsScreen::cnfg);
          break;

        case 4:
```

```
114:            auton.auton3(OptionsScreen::cnfg);
115:            break;
116:
117:          case 5:
118:            auton.auton4(OptionsScreen::cnfg);
119:            break;
120:
121:          case 6:
122:            auton.auton5(OptionsScreen::cnfg);
123:            break;
124:
125:          case 7:
126:            auton.auton6(OptionsScreen::cnfg);
127:            break;
128:
129:
130:
131:      }
132:
133:  }
134:
135:
136:  void wr( void* )
137:  {
138:      Writer writer;
139:      while ( 1 )
140:      {
141:          //std::cout << "dumping " << writer.get_count() << " items\n";
142:          //pros::delay(50);
143:          //std::cout << writer.get_count() << "\n";
144:          writer.dump();
145:          //std::cout << pros::millis() << "\n";
146:      }
147:  }
148:
149:
150:
151:  void Exit( int signal )
152:  {
153:      //Writer writer;
154:      std::cerr << "program caught " << signal << "\n" << std::flush;
155:      std::cerr << "errno: " << errno << "\n" << std::flush;
156:      std::cerr << "strerror: " << std::strerror(errno) << "\n" << std::flush;
157:      raise(signal);
158:  }
159:
160:
161:
162:  /**
163:   * Runs the operator control code. This function will be started in its own task
164:   * with the default priority and stack size whenever the robot is enabled via
165:   * the Field Management System or the VEX Competition Switch in the operator
166:   * control mode.
167:   *
168:   * If no competition control is connected, this function will run immediately
169:   * following initialize().
170:   *
171:   * If the robot is disabled or communications is lost, the
172:   * operator control task will be stopped. Re-enabling the robot will restart the
173:   * task, not resume it from where it left off.
174:   */
175:  void opcontrol() {
176:      std::cout << "opcontrol started\n";
177:
178:      std::signal(SIGSEGV, Exit);
179:      std::signal(SIGTERM, Exit);
180:      std::signal(SIGINT, Exit);
181:      std::signal(SIGILL, Exit);
182:      std::signal(SIGABRT, Exit);
183:      std::signal(SIGFPE, Exit);
184:      std::signal(SIGBUS, Exit);
185:      std::signal(SIGALRM, Exit);
186:      std::signal(SIGSTOP, Exit);
187:      std::signal(SIGUSR1, Exit);
188:      std::signal(SIGUSR2, Exit);
189:      std::signal(SIGKILL, Exit);
190:
191:      pros::delay(100);
192:
193:      lv_scr_load(tempScreen::temp_screen);
194:
195:      Motors *motors = Motors::get_instance(); //init singleton motors object
196:      motors->lift->set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
197:      motors->right_intake->set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
198:      motors->left_intake->set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
199:
200:      pros::Task driver_control_task (driver_control,
201:                                      (void*)NULL,
202:                                      TASK_PRIORITY_DEFAULT,
203:                                      TASK_STACK_DEPTH_DEFAULT,
204:                                      "DriverControlTask");
205:
206:  /*    pros::Task write_task (wr,
207:                          (void*)NULL,
208:                          TASK_PRIORITY_DEFAULT,
209:                          TASK_STACK_DEPTH_DEFAULT,
210:                          "write_task");
211:      std::cout << "starting\n";
212:      Motors motors;
213:      motors.record_macro();
214:
215:      Writer writer;
216:      while( writer.get_count() > 0 )
217:      {
218:          std::cout << pros::millis() << " " << writer.get_count() << "\n";
219:          pros::delay(1);
220:      }
221:
222:      std::cout << "done\n";*/
223:
224:
225:
226:      while(1)
```

```
227:     {
228:       Controller controllers;
229:       Motors *motors = Motors::get_instance();
230:       //partner button A runs autonomous
231:       if (controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_A))
232:       {
233:         motors->allow_left_chassis = false;
234:         motors->allow_right_chassis = false;
235:         motors->allow_intake = false;
236:         motors->allow_tilter = false;
237:         motors->allow_lift = false;
238:
239:         Autons auton;
240:         switch(final_auton_choice)
241:         {
242:           case 1:
243:             break;
244:
245:           case 2:
246:             std::cout << "starting auton\n";
247:             auton.auton1(OptionsScreen::cnfg);
248:             std::cout << "auton finished\n";
249:             break;
250:
251:           case 3:
252:             auton.auton2(OptionsScreen::cnfg);
253:             break;
254:
255:           case 4:
256:             auton.auton3(OptionsScreen::cnfg);
257:             break;
258:
259:           case 5:
260:             auton.auton4(OptionsScreen::cnfg);
261:             break;
262:
263:           case 6:
264:              auton.auton5(OptionsScreen::cnfg);
265:              break;
266:
267:           case 7:
268:             auton.auton6(OptionsScreen::cnfg);
269:              break;
270:
271:         }
272:       }
273:       pros::delay(20);
274:     }
275:   }
```

```
1:  /**
2:   * @file: ./RobotCode/src/Autons.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/5/19
5:   * @reviewed_by: Aiden Carney
6:   * TODO: add and test autons as well as add functionality for autons written on the sd card
7:   *
8:   * contains class that holds data about the autonomous period as well as
9:   * structs for configuration data
10:  */
11:
12: #ifndef __AUTONS_HPP__
13: #define __AUTONS_HPP__
14:
15: #include <unordered_map>
16:
17: #include "main.h"
18:
19:
20: typedef struct
21: {
22:     bool use_hardcoded = 0;
23:     bool gyro_turn = 1;
24:     bool accelleration_ctrl = 1;
25:     bool check_motor_tmp = 0;
26:     bool use_previous_macros = 1;
27:     bool record = 0;
28: } autonConfig;
29:
30:
31:
32: /**
33:  * @see: Motors.hpp
34:  * @see: ./objects/lcdCode
35:  *
36:  * contains data for the autonomous period as well as functions to run the
37:  * selected autonomous
38:  */
39: class Autons
40: {
41:     private:
42:
43:
44:     public:
45:         Autons();
46:         ~Autons();
47:
48:
49:         int debug_auton_num;       //change if more autons are added
50:                                    //debugger should be last option
51:         int driver_control_num;
52:
53:         const std::unordered_map <int, const char*> AUTONOMOUS_NAMES = {
54:             {1, "Driver Control"},          //used to find name of auton
55:             {2, "auton1"},                  //selected to keep title the same
56:             {3, "auton2"},
57:             {4, "auton3"},
58:             {5, "auton4"},
59:             {6, "auton5"},
60:             {7, "auton6"},
61:             {8, "Debugger"}
62:         };
63:         const std::unordered_map <int, const char*> AUTONOMOUS_DESCRIPTIONS = {   //used to find color of auton
64:             {1, "goes directly to driver control"},                     //selected to keep background the same
65:             {2, "scores four cubes in smallest\nzone"},
66:             {3, "scores four cubes in smallest\nzone"},
67:             {4, "scores in the big zone"},
68:             {5, "scores in the big zone"},
69:             {6, "drives forward and backwards"},
70:             {7, "runs a unit test so programmer\ncan understand what values should be"},
71:             {8, "opens debugger"}
72:         };
73:         const std::unordered_map <int, std::string> AUTONOMOUS_COLORS = {
74:             {1, "None"},                //used to find color of auton
75:             {2, "red"},                 //selected to keep background the same
76:             {3, "blue"},
77:             {4, "red"},
78:             {5, "blue"},
79:             {6, "None"},
80:             {7, "None"},
81:             {8, "None"}
82:         };
83:
84:
85:         /**
86:          * @param: autonConfig cnfg -> the configuration to use for the auton
87:          * @return: None
88:          *
89:          * @see: Motors.hpp
90:          *
91:          * scores four cubes in the smallest zone for the red team
92:          */
93:         void auton1( autonConfig cnfg );
94:
95:         /**
96:          * @param: autonConfig cnfg -> the configuration to use for the auton
97:          * @return: None
98:          *
99:          * @see: Motors.hpp
100:         *
101:         * scores four cubes in the smallest zone for the blue team
102:         */
103:        void auton2( autonConfig cnfg );
104:
105:
106:
107:        /**
108:         * @param: autonConfig cnfg -> the configuration to use for the auton
109:         * @return: None
110:         *
111:         * @see: Motors.hpp
112:         *
113:         * scores cubes in the big zone for red
```

```
114:         */
115:        void auton3( autonConfig cnfg );
116:
117:
118:
119:
120:        /**
121:         * @param: autonConfig cnfg -> the configuration to use for the auton
122:         * @return: None
123:         *
124:         * @see: Motors.hpp
125:         *
126:         * scores cubes in the big zone for blue
127:         */
128:        void auton4( autonConfig cnfg );
129:
130:
131:
132:
133:        /**
134:         * @param: autonConfig cnfg -> the configuration to use for the auton
135:         * @return: None
136:         *
137:         * @see: Motors.hpp
138:         *
139:         * drives forward
140:         */
141:        void auton5( autonConfig cnfg );
142:
143:
144:
145:
146:        /**
147:         * @param: autonConfig cnfg -> the configuration to use for the auton
148:         * @return: None
149:         *
150:         * @see: Motors.hpp
151:         *
152:         * runs a unit test
153:         */
154:        void auton6( autonConfig cnfg );
155:
156:
157:    };
158:
159:
160:
161:
162:    #endif
```

```
  1: /**
  2:  * @file: ./RobotCode/src/Autons.cpp
  3:  * @author: Aiden Carney
  4:  * @reviewed_on: 12/5/19
  5:  * @reviewed_by: Aiden Carney
  6:  *
  7:  * @see: Autons.hpp
  8:  *
  9:  * contains implementation for autonomous options
 10:  */
 11:
 12: #include <unordered_map>
 13:
 14: #include "main.h"
 15:
 16: #include "Autons.hpp"
 17: #include "objects/motors/Motors.hpp"
 18: #include "objects/robotChassis/chassis.hpp"
 19: #include "objects/tilter/Tilter.hpp"
 20: #include "objects/lift/Lift.hpp"
 21:
 22:
 23: Autons::Autons( )
 24: {
 25:     debug_auton_num = 8;
 26:     driver_control_num = 1;
 27: }
 28:
 29:
 30:
 31: Autons::~Autons( )
 32: {
 33:
 34: }
 35:
 36:
 37:
 38:
 39: /**
 40:  * scores four cubes in smaller zone for red side
 41:  */
 42: void Autons::auton1( autonConfig cnfg )
 43: {
 44:     Motors *motors = Motors::get_instance();
 45:     Tilter tilter;
 46:     Chassis chassis;
 47:     Lift lift;
 48:
 49:     //deploy
 50:     lift.move(360, 500);
 51:     motors->right_intake->move(-127);
 52:     motors->left_intake->move(-127);
 53:     pros::delay(700);
 54:     lift.move(-360, 500);
 55:
 56:     chassis.straight(440, 100);
 57:
 58:     for ( int i = 0; i < 4; i++ )
 59:     {
 60:         motors->right_intake->move(127);
 61:         motors->left_intake->move(127);
 62:         chassis.straight(160, 30);
 63:         pros::delay(50);
 64:         //pros::delay(150);
 65:     }
 66:
 67:     chassis.straight(50, 40);
 68:
 69:     motors->right_intake->move(80);
 70:     motors->left_intake->move(80);
 71:
 72:     chassis.straight(-500, 100);
 73:
 74:     chassis.turnRight(520); // 180 degrees
 75:
 76:     chassis.straight(400, 90);
 77:
 78:     chassis.turnLeft(200, 700); // 65 degrees
 79:
 80:     chassis.straight(200, 80, 500);
 81:
 82:     chassis.straight(100, 50, 500);
 83:
 84:     //dump stack
 85:     tilter.move(1200); //move stack upright
 86:
 87:     pros::delay(100);
 88:
 89:     motors->right_intake->move(-50); //outake to release
 90:     motors->left_intake->move(-50);
 91:
 92:     chassis.straight(-500, 30); //back up from stack
 93:
 94:     tilter.move(-900); //move stack upright
 95:
 96: }
 97:
 98:
 99:
100:
101: /**
102:  * scores four cubes in smaller zone for blue side
103:  */
104: void Autons::auton2( autonConfig cnfg )
105: {
106:     Motors *motors = Motors::get_instance();
107:     Tilter tilter;
108:     Chassis chassis;
109:     Lift lift;
110:
111:     //deploy
112:     lift.move(360, 500);
113:     motors->right_intake->move(-127);
```

```
114:     motors->left_intake->move(-127);
115:     pros::delay(700);
116:     lift.move(-360, 500);
117:
118:     chassis.straight(440, 100);
119:
120:     for ( int i = 0; i < 4; i++ )
121:     {
122:        motors->right_intake->move(127);
123:        motors->left_intake->move(127);
124:        chassis.straight(160, 30);
125:        pros::delay(50);
126:        //pros::delay(150);
127:     }
128:
129:     chassis.straight(50, 40);
130:
131:     motors->right_intake->move(80);
132:     motors->left_intake->move(80);
133:
134:     chassis.straight(-500, 100);
135:
136:     chassis.turnLeft(520); // 180 degrees
137:
138:     chassis.straight(400, 90);
139:
140:     chassis.turnRight(200, 700); // 65 degrees
141:
142:     chassis.straight(200, 80, 500);
143:
144:     chassis.straight(100, 50, 500);
145:
146:     //dump stack
147:     tilter.move(1200); //move stack upright
148:
149:     pros::delay(100);
150:
151:     motors->right_intake->move(-50); //outake to release
152:     motors->left_intake->move(-50);
153:
154:     chassis.straight(-500, 30); //back up from stack
155:
156:     tilter.move(-900); //move stack upright
157: }
158:
159:
160:
161:
162: /**
163:  * scores cubes in the big zone for red
164:  */
165: void Autons::auton3( autonConfig cnfg )
166: {
167:     Motors *motors = Motors::get_instance();
168:     Tilter tilter;
169:     Chassis chassis;
170:     Lift lift;
171:
172:     //push cube out of way
173:     chassis.straight(400, 75);
174:     chassis.straight(-200, 75);
175:
176:     //deploy
177:     lift.move(360, 500);
178:     motors->right_intake->move(-127);
179:     motors->left_intake->move(-127);
180:     pros::delay(1200);
181:     lift.move(-360, 500);
182:
183:     motors->right_intake->move(127);
184:     motors->left_intake->move(127);
185:
186:     chassis.straight(600, 100);
187:
188:     //move to right in front of cube and bring arms up
189:     chassis.straight(150, 100);
190:     lift.move(720, 800);
191:     chassis.straight(50, 50);
192:
193:     //move arms down while going forward
194:     lift.move(-720, 800);
195:     chassis.straight(350, 50);
196:
197:     //back up
198:     chassis.straight(-630, 100);
199:
200:     //turn towards cube
201:     chassis.turnLeft(270); //approx. 90 degrees
202:
203:     //drive up to cube
204:     chassis.straight(330, 100);
205:
206:     //pick up cube
207:     chassis.straight(340, 60);
208:
209:     //turn to zone
210:     chassis.turnLeft(150); //approx. 50 degrees
211:
212:     //move to zone
213:     chassis.straight(430, 75, 2000);
214:
215:     //dump stack
216:     tilter.move(1200); //move stack upright
217:
218:     pros::delay(100);
219:
220:     motors->right_intake->move(-50); //outake to release
221:     motors->left_intake->move(-50);
222:
223:     chassis.straight(-500, 30); //back up from stack
224:
225:     tilter.move(-900); //move stack upright
226:
```

```
227:  }
228:
229:
230:
231:
232:  /**
233:   * scores cubes in the big zone for blue
234:   */
235:  void Autons::auton4( autonConfig cnfg )
236:  {
237:    Motors *motors = Motors::get_instance();
238:    Tilter tilter;
239:    Chassis chassis;
240:    Lift lift;
241:
242:    //push cube out of way
243:    chassis.straight(400, 75);
244:    chassis.straight(-200, 75);
245:
246:    //deploy
247:    lift.move(360, 500);
248:    motors->right_intake->move(-127);
249:    motors->left_intake->move(-127);
250:    pros::delay(1200);
251:    lift.move(-360, 500);
252:
253:    motors->right_intake->move(127);
254:    motors->left_intake->move(127);
255:
256:    chassis.straight(600, 100);
257:
258:    //move to right in front of cube and bring arms up
259:    chassis.straight(150, 100);
260:    lift.move(720, 800);
261:    chassis.straight(50, 50);
262:
263:    //move arms down while going forward
264:    lift.move(-720, 800);
265:    chassis.straight(350, 50);
266:
267:    //back up
268:    chassis.straight(-630, 100);
269:
270:    //turn towards cube
271:    chassis.turnRight(270); //approx. 90 degrees
272:
273:    //drive up to cube
274:    chassis.straight(330, 100);
275:
276:    //pick up cube
277:    chassis.straight(340, 60);
278:
279:    //turn to zone
280:    chassis.turnRight(150); //approx. 50 degrees
281:
282:    //move to zone
283:    chassis.straight(430, 75, 2000);
284:
285:    //dump stack
286:    tilter.move(1200); //move stack upright
287:
288:    pros::delay(100);
289:
290:    motors->right_intake->move(-50); //outake to release
291:    motors->left_intake->move(-50);
292:
293:    chassis.straight(-500, 30); //back up from stack
294:
295:    tilter.move(-900); //move stack upright
296:  }
297:
298:
299:
300:
301:  /**
302:   * drives forward to score in the zone, then drive backward
303:   * to stop touching the cube
304:   */
305:  void Autons::auton5( autonConfig cnfg )
306:  {
307:    Chassis chassis;
308:    Motors *motors = Motors::get_instance();
309:    Lift lift;
310:
311:
312:    chassis.straight(-500, 100, 5000);
313:    pros::delay(500);
314:    chassis.straight(500, 100);
315:
316:    pros::delay(2000);
317:
318:    //deploy
319:    lift.move(360, 500);
320:    motors->right_intake->move(-127);
321:    motors->left_intake->move(-127);
322:    pros::delay(1200);
323:    lift.move(-450, 500);
324:  }
325:
326:
327:
328:
329:
330:  /**
331:   * runs unit test
332:   * 180 degree, 90 degree, 45 degree, 45 degree
333:   * tilter movement
334:   * straight drive moving
335:   */
336:  void Autons::auton6( autonConfig cnfg )
337:  {
338:    Motors *motors = Motors::get_instance();
339:    Tilter tilter;
```

```
340:     Chassis chassis;
341:
342:     //turn testing
343:
344:     chassis.turnRight(700); //180
345:     pros::delay(750);
346:     chassis.turnRight(350); //90
347:     pros::delay(750);
348:     chassis.turnRight(175); //45
349:     pros::delay(750);
350:     chassis.turnRight(175); //45
351:     pros::delay(750);
352:
353:     //tilter testing
354:     tilter.move(100);
355:     pros::delay(750);
356:     tilter.move(-100);
357:     pros::delay(750);
358:
359:     //straight drive testing
360:     chassis.straight(500, 100);
361:     pros::delay(750);
362:     chassis.straight(-500, 50);
363:     pros::delay(750);
364: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/Configuration.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on:
5:   * @reviewed_by:
6:   *
7:   * contains class static variables for runtime configuration
8:   *
9:   */
10:
11:  #ifndef __CONFIGURATION_HPP__
12:  #define __CONFIGURATION_HPP__
13:
14:  #include <iostream>
15:  #include <vector>
16:
17:  #include "main.h"
18:
19:  #include "../lib/json.hpp"
20:
21:
22:  typedef struct
23:  {
24:      double kP = 0;
25:      double kI = 0;
26:      double kD = 0;
27:      double I_max = 0;
28:      void print() {
29:          std::cout << "kP: " << this->kP << "\n";
30:          std::cout << "kI: " << this->kI << "\n";
31:          std::cout << "kD: " << this->kD << "\n";
32:          std::cout << "I_max: " << this->I_max << "\n";
33:      };
34:  } pid;
35:
36:
37:
38:  /**
39:   * @see: ../lib/json.hpp
40:   *
41:   * Singleton class
42:   * contains class to read data from config file on sd card for better runtime config
43:   * useful so that a clean build is not always necessary
44:   * contains static variables used throughout rest of project
45:   */
46:  class Configuration
47:  {
48:      private:
49:          Configuration();
50:          static Configuration *config_obj;
51:
52:      public:
53:          ~Configuration();
54:
55:          /**
56:           * @return: Configuration -> instance of class to be used throughout program
57:           *
58:           * give user the instance of the singleton class or creates it if it does
59:           * not yet exist
60:           */
61:          static Configuration* get_instance();
62:
63:          pid internal_motor_pid;
64:          pid tilter_pid_consts;
65:
66:          int front_right_port;
67:          int back_left_port;
68:          int front_left_port;
69:          int back_right_port;
70:          int left_intake_port;
71:          int right_intake_port;
72:          int tilter_port;
73:          int lift_port;
74:
75:          bool front_right_reversed;
76:          bool back_left_reversed;
77:          bool front_left_reversed;
78:          bool back_right_reversed;
79:          bool left_intake_reversed;
80:          bool right_intake_reversed;
81:          bool tilter_reversed;
82:          bool lift_reversed;
83:
84:          std::vector<int> lift_setpoints;
85:          std::vector<int> tilter_setpoints;
86:          std::vector<int> intake_speeds;
87:
88:
89:          /**
90:           * @return: int -> 1 if file was successfully read, 0 if no changes were made
91:           *
92:           * @see: ../lib/json.hpp
93:           *
94:           * parses json file looking for data to set variables to
95:           */
96:          int init();
97:
98:
99:          /**
100:          * @return: None
101:          *
102:          * @see: typedef struct pid
103:          *
104:          * prints all the variables in the class
105:          * used for debugging to make sure values are what they are
106:          * supposed to be
107:          */
108:         void print_config_options();
109:
110: };
111:
112:
113:
```

```
114:
115: #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/Configuration.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on:
5:   * @reviewed_by:
6:   *
7:   * @see: Configuration.hpp
8:   *
9:   * contains implementation for configuration class
10:  */
11:
12:  #include <vector>
13:  #include <fstream>
14:
15:  #include "main.h"
16:
17:  #include "../lib/json.hpp"
18:  #include "Configuration.hpp"
19:
20:
21:  Configuration *Configuration::config_obj = NULL;
22:
23:
24:  Configuration::Configuration( )
25:  {
26:    //set default values for constants in case file can't be read
27:    internal_motor_pid.kP = 1;
28:    internal_motor_pid.kI = 0;
29:    internal_motor_pid.kD = 0;
30:    internal_motor_pid.I_max = 0;
31:
32:    tilter_pid_consts.kP = 1;
33:    tilter_pid_consts.kI = 0;
34:    tilter_pid_consts.kD = 0;
35:    tilter_pid_consts.I_max = 0;
36:
37:    front_right_port = 9;
38:    back_left_port = 20;
39:    front_left_port = 10;
40:    back_right_port = 19;
41:    left_intake_port = 1;
42:    right_intake_port = 2;
43:    tilter_port = 15;
44:    lift_port = 16;
45:
46:    front_right_reversed = 1;
47:    back_left_reversed = 1;
48:    front_left_reversed = 1;
49:    back_right_reversed = 1;
50:    left_intake_reversed = 0;
51:    right_intake_reversed = 1;
52:    tilter_reversed = 1;
53:    lift_reversed = 0;
54:
55:    std::vector<int> vec1 {100, 300, 400, 500};
56:    std::vector<int> vec2 {100, 300, 400, 500};
57:    std::vector<int> vec3 {-63, -30, 0, 30, 63};
58:
59:    tilter_setpoints = vec1;
60:    lift_setpoints = vec2;
61:    intake_speeds = vec3;
62:  }
63:
64:
65:
66:  Configuration::~Configuration( )
67:  {
68:
69:  }
70:
71:
72:
73:  /**
74:   * inits object if object is not already initialized based on a static bool
75:   * sets bool if it is not set
76:   */
77:  Configuration* Configuration::get_instance()
78:  {
79:    if ( config_obj == NULL )
80:    {
81:      config_obj = new Configuration;
82:    }
83:    return config_obj;
84:  }
85:
86:
87:  /**
88:   * reads json file into memory in the form of a json object supported by
89:   * a library
90:   * parses json array to get pid constants and setpoints by looking at the size
91:   * sets other variables by looking at their value
92:   */
93:  int Configuration::init()
94:  {
95:    std::ifstream input("/usd/config.json"); //open file with library
96:    if ( input.fail() )
97:    {
98:      std::cerr << "[ERROR] " << pros::millis() << " configuration file could not be opened\n";
99:      return 0;
100:   }
101:   nlohmann::json contents;
102:   input >> contents;
103:
104:
105:   std::vector<double> constants1;  //read pid constants for different systems
106:   std::vector<double> constants2;
107:
108:   for ( int i1 = 0; i1 < 4; i1++)
109:   {
110:     double value1 = contents["internal_motor_pid"][i1];
111:     double value2 = contents["tilter_pid_consts"][i1];
112:
113:     std::cout << value1 << "\n";
```

```
114:        constants1.push_back(value1);
115:        constants2.push_back(value2);
116:    }
117:
118:    internal_motor_pid.kP = constants1.at(0);
119:    internal_motor_pid.kI = constants1.at(1);
120:    internal_motor_pid.kD = constants1.at(2);
121:    internal_motor_pid.I_max = constants1.at(3);
122:
123:    tilter_pid_consts.kP = constants2.at(0);
124:    tilter_pid_consts.kI = constants2.at(1);
125:    tilter_pid_consts.kD = constants2.at(2);
126:    tilter_pid_consts.I_max = constants2.at(3);
127:
128:
129:
130:
131:    front_right_port = contents["front_right_port"];  //read motor port definitions
132:    back_left_port = contents["back_left_port"];
133:    front_left_port = contents["front_left_port"];
134:    back_right_port = contents["back_right_port"];
135:    left_intake_port = contents["left_intake_port"];
136:    right_intake_port = contents["right_intake_port"];
137:    tilter_port = contents["tilter_port"];
138:    lift_port = contents["lift_port"];
139:
140:    front_right_reversed = contents["front_right_reversed"] == 1 ? true : false; //read motor port reversals
141:    back_left_reversed = contents["back_left_reversed"] == 1 ? true : false;
142:    front_left_reversed = contents["front_left_reversed"] == 1 ? true : false;
143:    back_right_reversed = contents["back_right_reversed"] == 1 ? true : false;
144:    left_intake_reversed = contents["left_intake_reversed"] == 1 ? true : false;
145:    right_intake_reversed = contents["right_intake_reversed"] == 1 ? true : false;
146:    tilter_reversed = contents["tilter_reversed"] == 1 ? true : false;
147:    lift_reversed = contents["lift_reversed"] == 1 ? true : false;
148:
149:
150:    tilter_setpoints.clear();
151:    for ( int i2 = 0; i2 < contents["tilter_setpoints"].size(); i2++)
152:    {
153:        tilter_setpoints.push_back(contents["tilter_setpoints"][i2]);
154:
155:    }
156:
157:    lift_setpoints.clear();
158:    for ( int i2 = 0; i2 < contents["lift_setpoints"].size(); i2++)
159:    {
160:        lift_setpoints.push_back(contents["lift_setpoints"][i2]);
161:
162:    }
163:
164:    intake_speeds.clear();
165:    for ( int i3 = 0; i3 < contents["intake_speeds"].size(); i3++)
166:    {
167:        intake_speeds.push_back(contents["intake_speeds"][i3]);
168:
169:    }
170: }
171:
172:
173:
174:
175: /**
176:  * prints all the variables and what they are so that they can be debugged
177:  * makes use of internal pid print function
178:  */
179: void Configuration::print_config_options()
180: {
181:    std::cout << "drive PID constants\n";
182:    internal_motor_pid.print();
183:    std::cout << "tilter PID constants\n";
184:    tilter_pid_consts.print();
185:
186:    std::cout << "\n";
187:
188:    std::cout << "front_right_port: " << front_right_port << "\n";
189:    std::cout << "back_left_port: " << back_left_port << "\n";
190:    std::cout << "front_left_port: " << front_left_port << "\n";
191:    std::cout << "back_right_port: " << back_right_port << "\n";
192:    std::cout << "left_intake_port: " << left_intake_port << "\n";
193:    std::cout << "right_intake_port: " << right_intake_port << "\n";
194:    std::cout << "tilter_port: " << tilter_port << "\n";
195:    std::cout << "lift_port: " << lift_port << "\n";
196:
197:    std::cout << "front_right_reversed: " << front_right_reversed << "\n";
198:    std::cout << "back_left_reversed: " << back_left_reversed << "\n";
199:    std::cout << "front_left_reversed: " << front_left_reversed << "\n";
200:    std::cout << "back_right_reversed: " << back_right_reversed << "\n";
201:    std::cout << "left_intake_reversed: " << left_intake_reversed << "\n";
202:    std::cout << "right_intake_reversed: " << right_intake_reversed << "\n";
203:    std::cout << "tilter_reversed: " << tilter_reversed << "\n";
204:    std::cout << "lift_reversed: " << lift_reversed << "\n";
205:
206:    std::cout << "\ntilter_setpoints: ";
207:    for ( int i = 0; i < tilter_setpoints.size() - 1; i++ )
208:    {
209:        std::cout << tilter_setpoints.at(i) << ", ";
210:    }
211:    std::cout << tilter_setpoints.at(tilter_setpoints.size() - 1) << "\n";
212:
213:
214:    std::cout << "\nlift_setpoints: ";
215:    for ( int i = 0; i < lift_setpoints.size() - 1; i++ )
216:    {
217:        std::cout << lift_setpoints.at(i) << ", ";
218:    }
219:    std::cout << lift_setpoints.at(lift_setpoints.size() - 1) << "\n";
220:
221:
222:    std::cout << "\nintake_speeds: ";
223:    for ( int i = 0; i < intake_speeds.size() - 1; i++ )
224:    {
225:        std::cout << intake_speeds.at(i) << ", ";
226:    }
```

```
227:     std::cout << intake_speeds.at(intake_speeds.size() - 1) << "\n";
228:   }
229:
```

```
1:  /**
2:   * @file: ./RobotCode/src/DriverControl.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: add more robot functions
7:   *
8:   * Contains robot move functions. Meant to be run in pros task
9:   *
10:  */
11:
12: #ifndef __DRIVERCONTROL_HPP__
13: #define __DRIVERCONTROL_HPP__
14:
15: #include <cstdlib>
16:
17: #include "../include/main.h"
18:
19: #include "objects/robotChassis/chassis.hpp"
20: #include "objects/controller/controller.hpp"
21: #include "objects/motors/Motors.hpp"
22:
23:
24:
25: /**
26:  * @param: void* -> not used
27:  * @return: None
28:  *
29:  * @see: Tilter.hpp
30:  *
31:  * gives the tilter holding strength for a set potentiometer reading
32:  * useful because the tilter is rubber banded and needs to be held backwards
33:  */
34: void tilter_holding_strength(void*);
35:
36:
37: /**
38:  * @param: void* -> not used
39:  * @return: None
40:  *
41:  * @see: Motors.hpp
42:  * @see: Controller.hpp
43:  *
44:  * meant to be run on task
45:  * function cycles through and allows user to controll robot
46:  *
47:  */
48: void driver_control(void*);
49:
50:
51: #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/DriverControl.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: DriverControl.hpp
8:   *
9:   */
10:
11:  #include <cstdlib>
12:  #include <cmath>
13:
14:  #include "../include/main.h"
15:
16:  #include "objects/robotChassis/chassis.hpp"
17:  #include "objects/controller/controller.hpp"
18:  #include "objects/motors/Motors.hpp"
19:  #include "objects/sensors/Sensors.hpp"
20:  #include "Configuration.hpp"
21:  #include "objects/tilter/Tilter.hpp"
22:  #include "DriverControl.hpp"
23:  #include "Configuration.hpp"
24:
25:
26:  /**
27:   * calls a blocking function for moving tilter to a set value
28:   * function will continuously be called while task is enabled
29:   */
30:  void tilter_holding_strength(void*)
31:  {
32:      Tilter tilter;
33:      while ( true )
34:      {
35:          tilter.move_to(100);
36:          pros::delay(20);
37:      }
38:  }
39:
40:
41:  /**
42:   * uses if statements to control motor based on controller settings
43:   * checks to set it to zero based on if static var in Motors class allows it
44:   * this is to make sure that other tasks can controll motors too
45:   */
46:  void driver_control(void*)
47:  {
48:      Controller controllers;
49:      Motors *motors = Motors::get_instance();
50:
51:      Chassis chassis;
52:      Tilter tilter;
53:      Sensors sensors;
54:
55:      Configuration *configuration = Configuration::get_instance();
56:
57:      bool hold_tilter = true;
58:      int intake_speed = 0;
59:
60:      bool right_x_axis_active = false;
61:      bool left_x_axis_active = false;
62:      bool move_tilter_home = false;
63:
64:      int down_arrow_last_click = 0;
65:
66:
67:      while ( true )
68:      {
69:          //master left analog y moves left side of robot
70:          if (
71:              (std::abs(controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y)) > 5 &&
72:              (std::abs(controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_X)) < 20))
73:              ||
74:              (std::abs(controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y)) > 20)
75:          )
76:          {
77:              float leftDriveSpeed = controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y);
78:              float corrected_speed = ( .000043326431866017 * std::pow( leftDriveSpeed, 3 ) ) + ( 0.29594689028631 * leftDriveSpeed);
79:              motors->frontLeft->move(corrected_speed);
80:              motors->backLeft->move(corrected_speed);
81:          }
82:          else if ( motors->allow_left_chassis && !right_x_axis_active && !left_x_axis_active  )
83:          {
84:              motors->frontLeft->move(0);
85:              motors->backLeft->move(0);
86:
87:          }
88:
89:          // uncomment for strafing functionality
90:          // //master left analog x moves strafes left and right
91:          // if (
92:          //     std::abs(controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_X)) > 20 &&
93:          //     std::abs(controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y)) < 5
94:          // )
95:          // {
96:          //     left_x_axis_active = true;
97:          //
98:          //     float drive_speed = controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_X);
99:          //     float corrected_speed = ( .000043326431866017 * std::pow( drive_speed, 3 ) ) + ( 0.29594689028631 * drive_speed);
100:         //
101:         //     //set up to strafe right, will strafe left when controller values are negative
102:         //     motors->frontRight->move(-drive_speed);
103:         //     motors->backRight->move(drive_speed);
104:         //     motors->frontLeft->move(drive_speed);
105:         //     motors->backLeft->move(-drive_speed);
106:         //
107:         //     motors->right_intake->move(intake_speed);
108:         //     motors->left_intake->move(intake_speed);
109:         //
110:         //
111:         // }
112:         // else
113:         // {
```

```
114:        //    left_x_axis_active = false;
115:        // }
116:
117:
118:
119:        //master right analog y moves right side of robot
120:        if (
121:           (std::abs(controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_Y)) > 5 &&
122:           (std::abs(controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X)) < 20))
123:           ||
124:           (std::abs(controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_Y)) > 20)
125:        )
126:        {
127:           float rightDriveSpeed = controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_Y);
128:           float corrected_speed = ( .000043326431866017 * std::pow( rightDriveSpeed, 3 ) ) + ( 0.29594689028631 * rightDriveSpeed);
129:           motors->frontRight->move(corrected_speed);
130:           motors->backRight->move(corrected_speed);
131:        }
132:        else if ( motors->allow_right_chassis && !right_x_axis_active && !left_x_axis_active )
133:        {
134:           motors->frontRight->move(0);
135:           motors->backRight->move(0);
136:
137:        }
138:
139:
140:        //master right analog x moves both sides of chassis slowly and outake slowly
141:        //or move forward and intake slowly
142:        if (
143:           controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X) > 20 &&
144:           std::abs(controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_Y)) < 5
145:        )
146:        {
147:           right_x_axis_active = true;
148:
149:           float drive_speed = controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X) / -5;
150:           float intake_speed = controllers.master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X) / -4;
151:
152:           motors->frontRight->move(drive_speed);
153:           motors->backRight->move(drive_speed);
154:           motors->frontLeft->move(drive_speed);
155:           motors->backLeft->move(drive_speed);
156:
157:           motors->right_intake->move(intake_speed);
158:           motors->left_intake->move(intake_speed);
159:
160:
161:        }
162:        else
163:        {
164:           right_x_axis_active = false;
165:        }
166:
167:
168:
169:        //master right digital moves the intake
170:        if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_R1) )
171:        {
172:           motors->right_intake->move(127);
173:           motors->left_intake->move(127);
174:        }
175:        else if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_R2) )
176:        {
177:           motors->right_intake->move(-127);
178:           motors->left_intake->move(-127);
179:        }
180:        // uncomment for intake always running functionality
181:        // else if ( intake_speed != 0 )
182:        // {
183:        //    motors->right_intake->move(intake_speed);
184:        //    motors->left_intake->move(intake_speed);
185:        // }
186:        // else if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_UP) )
187:        // {
188:        //    motors->right_intake->move(-40);
189:        //    motors->left_intake->move(-40);
190:        // }
191:        else if ( motors->allow_intake && !intake_speed && !right_x_axis_active )
192:        {
193:           motors->right_intake->move(0);
194:           motors->left_intake->move(0);
195:        }
196:
197:
198:        // uncomment for intake always running functionality
199:        // //master up and down arrows change the intake speed
200:        // if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_UP) )
201:        // {
202:        //    std::vector<int>::iterator elem = std::find (configuration->intake_speeds.begin(),
203:        //                                    configuration->intake_speeds.end(),
204:        //                                    intake_speed);
205:        //    int index = std::distance(configuration->intake_speeds.begin(), elem);
206:        //    index += 1;
207:        //    if ( index > configuration->intake_speeds.size() - 1 )  //cap speed
208:        //    {
209:        //       index = configuration->intake_speeds.size() - 1;
210:        //    }
211:        //    intake_speed = configuration->intake_speeds.at(index);
212:        //    pros::delay(200);  //add delay to make up for bad hardware
213:        // }
214:        // else if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_DOWN) )
215:        // {
216:        //    std::vector<int>::iterator elem = std::find (configuration->intake_speeds.begin(),
217:        //                                    configuration->intake_speeds.end(),
218:        //                                    intake_speed);
219:        //    int index = std::distance(configuration->intake_speeds.begin(), elem);
220:        //    index -= 1;
221:        //    if ( index < 0 )  //cap speed
222:        //    {
223:        //       index = 0;
224:        //    }
225:        //    intake_speed = configuration->intake_speeds.at(index);
226:        //    pros::delay(200);  //add delay to make up for bad hardware
```

```
227:        // }
228:
229:
230:        //master up arrow moves tilter forward
231:        if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_UP) )
232:        {
233:              motors->tilter->move(127);
234:        }
235:        // uncomment for auto move back functionality (requires a limit switch)
236:        // else if (
237:        //    controllers.master.get_digital_new_press(pros::E_CONTROLLER_DIGITAL_DOWN)
238:        //    && (pros::millis() - down_arrow_last_click) < 500)
239:        // {
240:        //    down_arrow_last_click = pros::millis();
241:        //    if ( !sensors.getLimitSwitch() )
242:        //    {
243:        //       motors->tilter->move(-127);
244:        //    }
245:        //    else
246:        //    {
247:        //       motors->tilter->move(0);
248:        //    }
249:        // }
250:        else if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_DOWN) )
251:        {
252:           motors->tilter->move(-127);
253:        }
254:        else if ( motors->allow_tilter && hold_tilter)
255:        {
256:           motors->tilter->move(0);
257:        }
258:
259:
260:        //master left digital moves the lift
261:        //partner right analog y moves the lift
262:        if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_L1) )
263:        {
264:           motors->lift->move(127);
265:        }
266:        else if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_L2) )
267:        {
268:           motors->lift->move(-127);
269:        }
270:        else if (std::abs(controllers.partner.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_Y)) > 0)
271:        {
272:           float lift_speed = controllers.partner.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_Y);
273:           float corrected_speed = ( .000043326431866017 * std::pow( lift_speed, 3 ) ) + ( 0.29594689028631 * lift_speed);
274:           motors->lift->move(corrected_speed);
275:        }
276:        else if ( motors->allow_lift && hold_tilter)
277:        {
278:           motors->lift->move(0);
279:        }
280:
281:
282:        //master button B cycles brakemode
283:        if (controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_B))
284:        {
285:           if ( motors->frontLeft->get_brake_mode() == pros::E_MOTOR_BRAKE_COAST )
286:           {
287:              motors->frontLeft->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
288:              motors->backLeft->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
289:              motors->frontRight->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
290:              motors->backRight->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
291:           }
292:           else
293:           {
294:              motors->frontLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
295:              motors->backLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
296:              motors->frontRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
297:              motors->backRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
298:           }
299:        }
300:
301:
302:        pros::delay(20);
303:     }
304:  }
```

```
1:  /**
2:   * @file: ./RobotCode/src/controller/controller.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 11/8/19
5:   * @reviewed_by: Aiden Carney
6:   * TODO: remove static memebers and replace with singleton class
7:   *
8:   * contains class that has data about controllers
9:   */
10:
11:  #include <unordered_map>
12:  #include <string>
13:
14:  #include "../../../include/main.h"
15:  #include "../../../include/api.h"
16:  #include "../../../include/pros/rtos.hpp"
17:  #include "../../../include/pros/motors.hpp"
18:
19:  #ifndef __CONTROLLER_HPP__
20:  #define __CONTROLLER_HPP__
21:
22:  /**
23:   * contains controller objects as well as unordered maps that hold information
24:   * about what the controller does
25:   * TODO: move unordere maps to configuration and make more robust so it does not have to be updated
26:   */
27:  class Controller
28:  {
29:      private:
30:
31:
32:      public:
33:          Controller();
34:          ~Controller();
35:
36:          static pros::Controller master;
37:          static pros::Controller partner;
38:
39:          static std::unordered_map <pros::controller_analog_e_t, std::string> MASTER_CONTROLLER_ANALOG_MAPPINGS;
40:          static std::unordered_map <pros::controller_analog_e_t, std::string> PARTNER_CONTROLLER_ANALOG_MAPPINGS;
41:          static std::unordered_map <pros::controller_digital_e_t, std::string> MASTER_CONTROLLER_DIGITAL_MAPPINGS;
42:          static std::unordered_map <pros::controller_digital_e_t, std::string>  PARTNER_CONTROLLER_DIGITAL_MAPPINGS;
43:
44:  };
45:
46:
47:
48:
49:  #endif
```

```
 1: /**
 2:  * @file: ./RobotCode/src/controller/controller.cpp
 3:  * @author: Aiden Carney
 4:  * @reviewed_on: 11/8/19
 5:  * @reviewed_by: Aiden Carney
 6:  *
 7:  * @see: controller.hpp
 8:  *
 9:  * contains definitions for static members of class
10:  */
11:
12: #include <unordered_map>
13: #include <string>
14:
15: #include "../../../include/main.h"
16: #include "../../../include/api.h"
17: #include "../../../include/pros/rtos.hpp"
18: #include "../../../include/pros/motors.hpp"
19:
20: #include "controller.hpp"
21:
22:
23:
24: pros::Controller Controller::master(pros::E_CONTROLLER_MASTER);
25: pros::Controller Controller::partner(pros::E_CONTROLLER_PARTNER);
26:
27: //mappings for each controller
28: std::unordered_map <pros::controller_analog_e_t, std::string> Controller::MASTER_CONTROLLER_ANALOG_MAPPINGS = {
29:     {pros::E_CONTROLLER_ANALOG_LEFT_X, "None"},
30:     {pros::E_CONTROLLER_ANALOG_LEFT_Y, "Left Side Chassis"},
31:     {pros::E_CONTROLLER_ANALOG_RIGHT_X, "None"},
32:     {pros::E_CONTROLLER_ANALOG_RIGHT_Y, "Right Side Chassis"}
33: };
34:
35: std::unordered_map <pros::controller_analog_e_t, std::string> Controller::PARTNER_CONTROLLER_ANALOG_MAPPINGS = {
36:     {pros::E_CONTROLLER_ANALOG_LEFT_X, "None"},
37:     {pros::E_CONTROLLER_ANALOG_LEFT_Y, "None"},
38:     {pros::E_CONTROLLER_ANALOG_RIGHT_X, "None"},
39:     {pros::E_CONTROLLER_ANALOG_RIGHT_Y, "None"}
40: };
41:
42: std::unordered_map <pros::controller_digital_e_t, std::string> Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS = {
43:     {pros::E_CONTROLLER_DIGITAL_L1, "Tilter Out"},
44:     {pros::E_CONTROLLER_DIGITAL_L2, "Tilter In"},
45:     {pros::E_CONTROLLER_DIGITAL_R2, "Outake"},
46:     {pros::E_CONTROLLER_DIGITAL_R1, "Intake"},
47:     {pros::E_CONTROLLER_DIGITAL_UP, "Intake constantly speed up"},
48:     {pros::E_CONTROLLER_DIGITAL_DOWN, "Intake constantly speed down"},
49:     {pros::E_CONTROLLER_DIGITAL_LEFT, "None"},
50:     {pros::E_CONTROLLER_DIGITAL_RIGHT, "None"},
51:     {pros::E_CONTROLLER_DIGITAL_X, "None"},
52:     {pros::E_CONTROLLER_DIGITAL_B, "Toggle brakes"},
53:     {pros::E_CONTROLLER_DIGITAL_Y, "None"},
54:     {pros::E_CONTROLLER_DIGITAL_A,  "Unit Test"}
55: };
56:
57: std::unordered_map <pros::controller_digital_e_t, std::string>  Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS = {
58:     {pros::E_CONTROLLER_DIGITAL_L1, "None"},
59:     {pros::E_CONTROLLER_DIGITAL_L2, "None"},
60:     {pros::E_CONTROLLER_DIGITAL_R2, "None"},
61:     {pros::E_CONTROLLER_DIGITAL_R1, "None"},
62:     {pros::E_CONTROLLER_DIGITAL_UP, "None"},
63:     {pros::E_CONTROLLER_DIGITAL_DOWN, "None"},
64:     {pros::E_CONTROLLER_DIGITAL_LEFT, "None"},
65:     {pros::E_CONTROLLER_DIGITAL_RIGHT, "None"},
66:     {pros::E_CONTROLLER_DIGITAL_X, "None"},
67:     {pros::E_CONTROLLER_DIGITAL_B, "None"},
68:     {pros::E_CONTROLLER_DIGITAL_Y, "None"},
69:     {pros::E_CONTROLLER_DIGITAL_A,  "None"}
70: };
71:
72:
73:
74:
75: Controller::Controller()
76: {
77:
78: }
79:
80: Controller::~Controller()
81: {
82:
83: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/motors/Motor.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on:
5:   * @reviewed_by:
6:   * TODO:
7:   *
8:   * contains a wrapper class for a pros::Motor
9:   */
10:
11:  #ifndef __MOTOR_HPP__
12:  #define __MOTOR_HPP__
13:
14:  #include <atomic>
15:
16:  #include "main.h"
17:
18:  #include "../../Configuration.hpp"
19:
20:
21:
22:  /**
23:   * @see: pros::Motor
24:   * @see: ../../Configuration.hpp
25:   *
26:   * wrapper class for pros::Motor
27:   * contains implementation for better runtime port configuration
28:   * contains easier implementation for slew rate control
29:   * contains a pid velocity controller that can be enabled for consistent motor output
30:   */
31:  class Motor
32:  {
33:      private:
34:          int motor_port;
35:
36:          pros::Motor *motor;
37:
38:          int log_level;
39:
40:          bool slew_enabled;
41:          int slew_rate;
42:
43:          bool velocity_pid_enabled;
44:          int prev_velocity;
45:          pid internal_motor_pid;
46:          double integral;
47:          double prev_error;
48:
49:          int prev_target_voltage;
50:          int target_voltage;
51:
52:          /**
53:           * @param int target -> the new voltage that could be requested
54:           * @param int previous -> the previous voltage to calculate change in voltage over time
55:           * @param in delta_t -> the time that has elapsed
56:           * @return: int -> the rate of the voltage set based on time elapsed and previous voltage
57:           *
58:           * @see: set_voltage()
59:           *
60:           * calculates the rate of change of the voltage (mv/ms) that the new
61:           * voltage is trying to reach
62:           */
63:          int calc_target_rate( int target, int previous, int delta_t );
64:
65:          /**
66:           * @param: int voltage -> a possible motor voltage in mv on interval [-12000,12000]
67:           * @return: int -> the corresponding velocity for a given voltage
68:           *
69:           * TODO: add checking if the voltage is not on the interval
70:           *
71:           * calculates the corresponding velocity for a given voltage in mv
72:           * the velocity range corresponds to the gearset of the motor
73:           * velocity ranges are ~20% higher than what they are rated for
74:           * because motors can achieve this velocity when supplied 12V
75:           */
76:          int calc_target_velocity( int voltage );
77:
78:          /**
79:           * @return: int -> the voltage that the motor will be set at
80:           *
81:           * @see: slew rate functions contained in this class
82:           *
83:           * returns the target voltage based on the voltage set by the user
84:           * but that is either increased or decreased by the velocity pid if that
85:           * is enabled, or the slew rate code which limits the rate that the
86:           * voltage can increase
87:           */
88:          int get_target_voltage( int delta_t );
89:
90:
91:          std::atomic<bool> lock;  //protect motor functions from concurrent access
92:          bool allow_driver_control;
93:
94:
95:      public:
96:          Motor(int port, pros::motor_gearset_e_t gearset, bool reversed);
97:          Motor(int port, pros::motor_gearset_e_t gearset, bool reversed, pid pid_consts);
98:          ~Motor();
99:
100:     //accessor functions
101:
102:         /**
103:          * @return: double -> the actual velocity of the motor
104:          *
105:          * @see: pros::Motor
106:          *
107:          * returns the actual velocity of the motor as calculated internally by
108:          * the pros::Motor
109:          */
110:         double get_actual_velocity( );
111:
112:         /**
113:          * @return: double -> the actual voltage of the motor
```

```
114:         *
115:         * @see: pros::Motor
116:         *
117:         * returns the actual voltage of the motor as calculated internally by
118:         * the pros::Motor
119:         */
120:        double get_actual_voltage( );
121:
122:        /**
123:         * @return: int -> the actual current being supplied to the motor
124:         *
125:         * @see: pros::Motor
126:         *
127:         * returns the actual current being supplied to the motor as calculated internally by
128:         * the pros::Motor
129:         */
130:        int get_current_draw( );
131:
132:        /**
133:         * @return: double -> the encoder value of the motor
134:         *
135:         * @see: pros::Motor
136:         *
137:         * returns the encoder position of the motor in degrees as calculated internally by
138:         * the pros::Motor
139:         */
140:        double get_encoder_position( );
141:
142:        /**
143:         * @return: pros::motor_gearset_e_t -> the gearing of the motor
144:         *
145:         * @see: pros::Motor
146:         *
147:         * returns the gearset internally used by the motor per the pros::Motor
148:         */
149:        pros::motor_gearset_e_t get_gearset( );
150:
151:        /**
152:         * @return: pros::motor_brake_mode_e_t -> the brakemode of the motor
153:         *
154:         * @see: pros::Motor
155:         *
156:         * returns the brakemode internally used by the motor per the pros::Motor
157:         */
158:        pros::motor_brake_mode_e_t get_brake_mode( );
159:
160:        /**
161:         * @return: int -> the port of the motor
162:         *
163:         * returns the port that the motor is set on
164:         */
165:        int get_port( );
166:
167:        /**
168:         * @return: pid -> struct of pid constants
169:         *
170:         * returns the pid constants in use by the motor
171:         */
172:        pid get_pid( );
173:
174:        /**
175:         * @return: int -> the slew rate in use by the motor
176:         *
177:         * returns the slew rate in mV/ms in use by the motor
178:         */
179:        int get_slew_rate( );
180:
181:        /**
182:         * @return: double -> the power drawn by the motor
183:         *
184:         * @see: pros::Motor
185:         *
186:         * returns the power that the motor is drawing in Watts
187:         */
188:        double get_power( );
189:
190:        /**
191:         * @return: double -> the temperature of the motor
192:         *
193:         * @see: pros::Motor
194:         *
195:         * returns the temperature of the motor in degrees C
196:         */
197:        double get_temperature( );
198:
199:        /**
200:         * @return: double -> the torque output of the motor
201:         *
202:         * @see: pros::Motor
203:         *
204:         * returns the torque output of the motor in Nm
205:         */
206:        double get_torque( );
207:
208:        /**
209:         * @return: int -> the direction the motor is spinning
210:         *
211:         * @see: pros::Motor
212:         *
213:         * returns the direction of the motor
214:         * 1 for moving in the positive direction
215:         * -1 for moving in the negative direction
216:         */
217:        int get_direction( );
218:
219:        /**
220:         * @return: int -> the efficiency of the motor
221:         *
222:         * @see: pros::Motor
223:         *
224:         * returns the efficiency of the motor as a percentage
225:         */
226:        int get_efficiency( );
```

```
227:
228:        /**
229:         * @return: int -> if the motor is a rest
230:         *
231:         * @see: pros::Motor
232:         *
233:         * returns 1 if the motor is not moving and 0 if the motor is moving
234:         */
235:        int is_stopped( );
236:
237:        /**
238:         * @return: int -> if the motor has been reversed or not
239:         *
240:         * @see: pros::Motor
241:         *
242:         * returns 1 if the motor has been reversed and 0 if the motor was not reversed
243:         */
244:        int is_reversed( );
245:
246:
247:
248:
249:    //setter functions
250:        int set_port( int port );
251:        int tare_encoder( );
252:        int set_brake_mode( pros::motor_brake_mode_e_t brake_mode );
253:        int set_gearing( pros::motor_gearset_e_t gearset );
254:        int reverse_motor( );
255:        int set_pid( pid pid_consts );
256:        void set_log_level( int logging );
257:
258:
259:    //movement functions
260:        int move( int voltage  );
261:        int move_velocity( int velocity );
262:        int set_voltage( int voltage );
263:
264:    //slew rate control functions
265:        int set_slew( int rate );
266:        void enable_slew( );
267:        void disable_slew( );
268:
269:
270:    //velocity pid control functions
271:        void enable_velocity_pid( );
272:        void disable_velocity_pid( );
273:
274:
275:    //driver control lock setting and clearing functions
276:        void enable_driver_control( );
277:        void disable_driver_control( );
278:        int driver_control_allowed( );
279:
280:
281:    //function to run on thread
282:        int run( int delta_t );
283:  };
284:
285:
286:
287:
288:  #endif
```

```
 1:
 2: #include "main.h"
 3:
 4: #include "../../Configuration.hpp"
 5: #include "Motor.hpp"
 6:
 7:
 8:
 9: Motor::Motor( int port, pros::motor_gearset_e_t gearset, bool reversed )
10: {
11:     lock = ATOMIC_VAR_INIT(false);
12:     allow_driver_control = true;
13:
14:     while ( lock.exchange( true ) ); //aquire motor lock
15:
16:     motor_port = port;
17:
18:     motor = new pros::Motor(port, gearset, reversed, pros::E_MOTOR_ENCODER_DEGREES);
19:
20:     velocity_pid_enabled = true; //default motor velocity pid controller
21:     prev_velocity = 0;
22:
23:     log_level = 0;
24:
25:     slew_enabled = true;
26:     slew_rate = 30;  //approx. 5% voltage per 20ms == 400ms to reach full voltage
27:
28:     prev_target_voltage = 0;
29:     target_voltage = 0;
30:
31:     Configuration *configuration = Configuration::get_instance();
32:     internal_motor_pid.kP = configuration->internal_motor_pid.kP;
33:     internal_motor_pid.kI = configuration->internal_motor_pid.kI;
34:     internal_motor_pid.kD = configuration->internal_motor_pid.kD;
35:     internal_motor_pid.I_max = configuration->internal_motor_pid.I_max;
36:     integral = 0;
37:     prev_error = 0;
38:
39:     lock.exchange(false);
40: }
41:
42:
43: Motor::Motor(int port, pros::motor_gearset_e_t gearset, bool reversed, pid pid_consts)
44: {
45:     lock = ATOMIC_VAR_INIT(false);
46:     allow_driver_control = true;
47:
48:     while ( lock.exchange( true ) ); //aquire motor lock
49:
50:     motor_port = port;
51:
52:     motor = new pros::Motor(port, gearset, reversed, pros::E_MOTOR_ENCODER_DEGREES);
53:
54:     velocity_pid_enabled = true; //default motor velocity pid controller
55:     prev_velocity = 0;
56:
57:     log_level = 0;
58:
59:     slew_enabled = true;
60:     slew_rate = 30;  //approx. 5% voltage per 20ms == 400ms to reach full voltage
61:
62:     prev_target_voltage = 0;
63:     target_voltage = 0;
64:
65:     internal_motor_pid.kP = pid_consts.kP;
66:     internal_motor_pid.kI = pid_consts.kI;
67:     internal_motor_pid.kD = pid_consts.kD;
68:     internal_motor_pid.I_max = pid_consts.I_max;
69:     integral = 0;
70:     prev_error = 0;
71:
72:     lock.exchange(false);
73: }
74:
75:
76: Motor::~Motor( )
77: {
78:     delete motor;
79: }
80:
81:
82:
83:
84: int Motor::calc_target_rate( int target, int previous, int delta_t)
85: {
86:     int delta_v = target - previous;
87:     int rate;
88:     if ( delta_t == 0 && delta_v == 0 )
89:     {
90:         rate = 0;
91:     }
92:     else if ( delta_t == 0 && delta_v != 0 )
93:     {
94:         rate = INT32_MAX;  //essentially undefined but still represented as integer
95:     }
96:     else
97:     {
98:         rate = delta_v / delta_t;
99:     }
100:
101:     return rate;
102: }
103:
104:
105: int Motor::calc_target_velocity( int voltage )
106: {
107:     int prev_max = 12000;
108:     int prev_min = -12000;
109:
110:     pros::motor_gearset_e_t gearset = motor->get_gearing();
111:
112:     int new_max;
113:     int new_min;
```

```
114:
115:    if ( gearset == pros::E_MOTOR_GEARSET_36 )  //100 RPM Motor
116:    {
117:       new_max = 120;
118:       new_min = -120;
119:    }
120:    if ( gearset == pros::E_MOTOR_GEARSET_06 ) //600 RPM Motor
121:    {
122:       new_max = 720;
123:       new_min = -720;
124:    }
125:    else //default to 200 RPM motor because that is most commonly used
126:    {
127:       new_max = 240;
128:       new_min = -240;
129:    }
130:
131:    int velocity = (((voltage - prev_min) * (new_max - new_min)) / (prev_max - prev_min)) + new_min;
132:
133:
134:    return velocity;
135: }
136:
137:
138: int Motor::get_target_voltage( int delta_t )
139: {
140:    double kP = internal_motor_pid.kP;
141:    double kI = internal_motor_pid.kI;
142:    double kD = internal_motor_pid.kD;
143:    double I_max = internal_motor_pid.I_max;
144:
145:    //int voltage = get_actual_voltage();
146:    int voltage;
147:    int calculated_target_voltage = target_voltage;
148:
149:    //velocity pid is enabled when the target voltage does not change
150:    if ( velocity_pid_enabled && target_voltage == prev_target_voltage )
151:    {
152:       int error =  calc_target_velocity(target_voltage) - get_actual_velocity();
153:       if ( integral == 0 || std::abs(integral) > I_max)
154:       {
155:          integral = 0;
156:       }
157:       else
158:       {
159:          integral = integral + error;
160:       }
161:       double derivative = error - prev_error;
162:       prev_error = error;
163:
164:       calculated_target_voltage = kP * error + kI * integral + kD * derivative;
165:
166:    }
167:
168:    //ensure that voltage range is allowed by the slew rate set
169:    int rate = calc_target_rate(calculated_target_voltage, prev_target_voltage, delta_t);
170:    if ( slew_enabled && std::abs(rate) > slew_rate )
171:    {
172:       int max_delta_v = slew_rate * delta_t;
173:
174:       int polarity = 1;              // rate will be positive or negative if motor is gaining
175:       if ( rate < 0 )                // or losing velocity
176:       {                           // the polarity ensures that the max voltage is added
177:          polarity = -1;           // in the correct direction so that the motor's velocity
178:       }                           // will increase in the correct direction
179:
180:       voltage = prev_target_voltage + (polarity * max_delta_v);
181:    }
182:    else
183:    {
184:       voltage = calculated_target_voltage;
185:    }
186:
187:
188:    return voltage;
189: }
190:
191:
192:
193:
194: //accessor functions
195: double Motor::get_actual_velocity( )
196: {
197:    return motor->get_actual_velocity();
198: }
199:
200:
201: double Motor::get_actual_voltage( )
202: {
203:    return motor->get_voltage();
204: }
205:
206:
207: int Motor::get_current_draw( )
208: {
209:    return motor->get_current_draw();
210: }
211:
212:
213: double Motor::get_encoder_position( )
214: {
215:    return motor->get_position();
216: }
217:
218:
219: pros::motor_gearset_e_t Motor::get_gearset( )
220: {
221:    return motor->get_gearing();
222: }
223:
224:
225: pros::motor_brake_mode_e_t Motor::get_brake_mode( )
226: {
```

```
227:     return motor->get_brake_mode();
228: }
229:
230:
231: int Motor::get_port( )
232: {
233:     return motor_port;
234: }
235:
236:
237: pid Motor::get_pid( )
238: {
239:     return internal_motor_pid;
240: }
241:
242:
243: int Motor::get_slew_rate( )
244: {
245:     return slew_rate;
246: }
247:
248:
249: double Motor::get_power( )
250: {
251:     return motor->get_power();
252: }
253:
254:
255: double Motor::get_temperature( )
256: {
257:     return motor->get_temperature();
258: }
259:
260:
261: double Motor::get_torque( )
262: {
263:     return motor->get_torque();
264: }
265:
266:
267: int Motor::get_direction( )
268: {
269:     return motor->get_direction();
270: }
271:
272:
273: int Motor::get_efficiency( )
274: {
275:     return motor->get_efficiency();
276: }
277:
278:
279: int Motor::is_stopped( )
280: {
281:     return motor->is_stopped();
282: }
283:
284:
285: int Motor::is_reversed( )
286: {
287:     return motor->is_reversed();
288: }
289:
290:
291:
292:
293:
294: //setter functions
295: int Motor::set_port( int port )
296: {
297:     pros::motor_gearset_e_t gearset = motor->get_gearing();
298:     bool reversed = motor->is_reversed();
299:
300:     while ( lock.exchange( true ) );
301:
302:     try
303:     {
304:         delete motor;
305:         motor = new pros::Motor(port, gearset, reversed, pros::E_MOTOR_ENCODER_DEGREES);
306:         motor_port = port;
307:     }
308:     catch(...) //ensure lock will be released
309:     {
310:         std::cerr << "[ERROR] " << pros::millis() << "could not set port on motor port " << motor_port << "\n";
311:         lock.exchange(false);
312:         return 0;
313:     }
314:
315:     lock.exchange(false);
316:     return 1;
317: }
318:
319:
320: int Motor::tare_encoder( )
321: {
322:     while ( lock.exchange( true ) );
323:
324:     try
325:     {
326:         motor->tare_position();
327:     }
328:     catch(...) //ensure lock will be released
329:     {
330:         std::cerr << "[ERROR] " << pros::millis() << "could not tare encoder on motor port " << motor_port << "\n";
331:         lock.exchange(false);
332:         return 0;
333:     }
334:
335:     lock.exchange(false);
336:
337:     return 1;
338: }
339:
```

```
340:
341:  int Motor::set_brake_mode( pros::motor_brake_mode_e_t brake_mode )
342:  {
343:      while ( lock.exchange( true ) );
344:
345:      try
346:      {
347:          motor->set_brake_mode(brake_mode);
348:      }
349:      catch(...) //ensure lock will be released
350:      {
351:          std::cerr << "[ERROR] " << pros::millis() << "could not set brakemode on motor port " << motor_port << "\n";
352:          lock.exchange(false);
353:          return 0;
354:      }
355:
356:      lock.exchange(false);
357:
358:      return 1;
359:  }
360:
361:
362:  int Motor::set_gearing( pros::motor_gearset_e_t gearset )
363:  {
364:      while ( lock.exchange( true ) );
365:
366:      try
367:      {
368:          motor->set_gearing(gearset);
369:      }
370:      catch(...) //ensure lock will be released
371:      {
372:          std::cerr << "[ERROR] " << pros::millis() << "could not set gearing on motor port " << motor_port << "\n";
373:          lock.exchange(false);
374:          return 0;
375:      }
376:
377:      lock.exchange(false);
378:
379:      return 1;
380:  }
381:
382:
383:  int Motor::reverse_motor( )
384:  {
385:      while ( lock.exchange( true ) );
386:
387:      try
388:      {
389:          motor->set_reversed(!motor->is_reversed());
390:      }
391:      catch(...) //ensure lock will be released
392:      {
393:          std::cerr << "[ERROR] " << pros::millis() << "could not reverse motor on port " << motor_port << "\n";
394:          lock.exchange(false);
395:          return 0;
396:      }
397:
398:      lock.exchange(false);
399:
400:      return 1;
401:  }
402:
403:
404:  int Motor::set_pid( pid pid_consts )
405:  {
406:      while ( lock.exchange( true ) );
407:
408:      try
409:      {
410:          internal_motor_pid.kP = pid_consts.kP;
411:          internal_motor_pid.kI = pid_consts.kI;
412:          internal_motor_pid.kD = pid_consts.kD;
413:          internal_motor_pid.I_max = pid_consts.I_max;
414:      }
415:      catch(...) //ensure lock will be released
416:      {
417:          std::cerr << "[ERROR] " << pros::millis() << "could not set motor pid on motor port " << motor_port << "\n";
418:          lock.exchange(false);
419:          return 0;
420:      }
421:
422:      lock.exchange(false);
423:
424:      return 1;
425:  }
426:
427:
428:  void Motor::set_log_level( int logging )
429:  {
430:      if ( logging > 5 )
431:      {
432:          log_level = 5;
433:      }
434:      else if ( logging < 0 )
435:      {
436:          log_level = 0;
437:      }
438:      else
439:      {
440:          log_level = logging;
441:      }
442:  }
443:
444:
445:
446:
447:  //movement functions
448:  int Motor::move( int voltage  )
449:  {
450:      int prev_max = 127;
451:      int prev_min = -127;
452:      int new_max = 12000;
```

```
453:     int new_min = -12000;
454:
455:     int scaled_voltage = (((voltage - prev_min) * (new_max - new_min)) / (prev_max - prev_min)) + new_min;
456:
457:     set_voltage(scaled_voltage); //dont aquire lock because it will be acquired in this function
458:
459:     return 1;
460:   }
461:
462:
463:   int Motor::move_velocity( int velocity )
464:   {
465:     pros::motor_gearset_e_t gearset = motor->get_gearing();
466:
467:     int prev_max;
468:     int prev_min;
469:
470:     if ( gearset == pros::E_MOTOR_GEARSET_36 )  //100 RPM Motor
471:     {
472:       prev_max = 120;
473:       prev_min = -120;
474:     }
475:     if ( gearset == pros::E_MOTOR_GEARSET_06 ) //600 RPM Motor
476:     {
477:       prev_max = 720;
478:       prev_min = -720;
479:     }
480:     else //default to 200 RPM motor because that is most commonly used
481:     {
482:       prev_max = 240;
483:       prev_min = -240;
484:     }
485:
486:     int new_max = 12000;
487:     int new_min = -12000;
488:
489:     int voltage = (((velocity - prev_min) * (new_max - new_min)) / (prev_max - prev_min)) + new_min;
490:
491:     set_voltage(voltage); //dont aquire lock because it will be acquired in this function
492:
493:     return 1;
494:   }
495:
496:
497:   int Motor::set_voltage( int voltage )
498:   {
499:     while ( lock.exchange( true ) );
500:     target_voltage = voltage;
501:     if ( target_voltage != prev_target_voltage )  //reset integral for new setpoint
502:     {
503:       integral = 0;
504:     }
505:     lock.exchange(false);
506:
507:     return 1;
508:   }
509:
510:
511:
512:
513:   //velocity pid control functions
514:   void Motor::enable_velocity_pid( )
515:   {
516:     while ( lock.exchange( true ) );
517:     velocity_pid_enabled = true;
518:     lock.exchange(false);
519:   }
520:
521:
522:   void Motor::disable_velocity_pid( )
523:   {
524:     while ( lock.exchange( true ) );
525:     velocity_pid_enabled = false;
526:     lock.exchange(false);
527:   }
528:
529:
530:
531:
532:   //slew control functions
533:   int Motor::set_slew( int rate )
534:   {
535:     while ( lock.exchange( true ) );
536:     slew_rate = rate;
537:     lock.exchange(false);
538:
539:     return 1;
540:   }
541:
542:
543:   void Motor::enable_slew( )
544:   {
545:     while ( lock.exchange( true ) );
546:     slew_enabled = true;
547:     lock.exchange(false);
548:   }
549:
550:
551:   void Motor::disable_slew( )
552:   {
553:     while ( lock.exchange( true ) );
554:     slew_enabled = false;
555:     lock.exchange(false);
556:   }
557:
558:
559:
560:
561:   //driver control lock setting and clearing functions
562:   void Motor::enable_driver_control()
563:   {
564:     while ( lock.exchange( true ) );
565:     allow_driver_control = true;
```

```
566:     lock.exchange(false);
567:   }
568:
569:
570:   void Motor::disable_driver_control()
571:   {
572:     while ( lock.exchange( true ) );
573:     allow_driver_control = false;
574:     lock.exchange(false);
575:   }
576:
577:
578:   int Motor::driver_control_allowed()
579:   {
580:     if ( allow_driver_control )
581:     {
582:       return 1;
583:     }
584:     else
585:     {
586:       return 0;
587:     }
588:   }
589:
590:
591:
592:
593:   int Motor::run( int delta_t )
594:   {
595:     int voltage = get_target_voltage( delta_t );
596:     motor->move_voltage(voltage);
597:
598:
599:     std::string log_msg;
600:     switch ( log_level )
601:     {
602:       case 0:
603:         log_msg = "";
604:         break;
605:
606:       case 1:
607:         log_msg = (
608:           "[INFO]" + std::string(" Motor ") + std::to_string(motor_port)
609:           + ", Actual_Vol: " + std::to_string(get_actual_voltage())
610:           + ", Brake: " + std::to_string(get_brake_mode())
611:           + ", Gear: " + std::to_string(get_gearset())
612:           + ", I_max: " + std::to_string(internal_motor_pid.I_max)
613:           + ", I: " + std::to_string(integral)
614:           + ", kD: " + std::to_string(internal_motor_pid.kD)
615:           + ", kI: " + std::to_string(internal_motor_pid.kI)
616:           + ", kP: " + std::to_string(internal_motor_pid.kP)
617:           + ", Slew: " + std::to_string(get_slew_rate())
618:           + ", Time: " + std::to_string(pros::millis())
619:           + ", Vel_Sp: " + std::to_string(calc_target_velocity(target_voltage))
620:           + ", Vel: " + std::to_string(get_actual_velocity())
621:         );
622:         break;
623:
624:       case 2:
625:         log_msg = (
626:           "[INFO]" + std::string(" Motor ") + std::to_string(motor_port)
627:           + ", Actual_Vol: " + std::to_string(get_actual_voltage())
628:           + ", Brake: " + std::to_string(get_brake_mode())
629:           + ", Calc_Target_Vol: " + std::to_string(voltage)
630:           + ", Gear: " + std::to_string(get_gearset())
631:           + ", I_max: " + std::to_string(internal_motor_pid.I_max)
632:           + ", I: " + std::to_string(integral)
633:           + ", kD: " + std::to_string(internal_motor_pid.kD)
634:           + ", kI: " + std::to_string(internal_motor_pid.kI)
635:           + ", kP: " + std::to_string(internal_motor_pid.kP)
636:           + ", Slew: " + std::to_string(get_slew_rate())
637:           + ", Target_Vol: " + std::to_string(target_voltage)
638:           + ", Time: " + std::to_string(pros::millis())
639:           + ", Vel_Sp: " + std::to_string(calc_target_velocity(target_voltage))
640:           + ", Vel: " + std::to_string(get_actual_velocity())
641:         );
642:         break;
643:
644:       case 3:
645:         log_msg = (
646:           "[INFO]" + std::string(" Motor ") + std::to_string(motor_port)
647:           + ", Actual_Vol: " + std::to_string(get_actual_voltage())
648:           + ", Brake: " + std::to_string(get_brake_mode())
649:           + ", Calc_Target_Vol: " + std::to_string(voltage)
650:           + ", Gear: " + std::to_string(get_gearset())
651:           + ", I_max: " + std::to_string(internal_motor_pid.I_max)
652:           + ", I: " + std::to_string(integral)
653:           + ", IME: " + std::to_string(get_encoder_position())
654:           + ", kD: " + std::to_string(internal_motor_pid.kD)
655:           + ", kI: " + std::to_string(internal_motor_pid.kI)
656:           + ", kP: " + std::to_string(internal_motor_pid.kP)
657:           + ", Slew: " + std::to_string(get_slew_rate())
658:           + ", Target_Vol: " + std::to_string(target_voltage)
659:           + ", Time: " + std::to_string(pros::millis())
660:           + ", Vel_Sp: " + std::to_string(calc_target_velocity(target_voltage))
661:           + ", Vel: " + std::to_string(get_actual_velocity())
662:         );
663:         break;
664:
665:       case 4:
666:         log_msg = (
667:           "[INFO]" + std::string(" Motor ") + std::to_string(motor_port)
668:           + ", Actual_Vol: " + std::to_string(get_actual_voltage())
669:           + ", Brake: " + std::to_string(get_brake_mode())
670:           + ", Calc_Target_Vol: " + std::to_string(voltage)
671:           + ", Dir: " + std::to_string(get_direction())
672:           + ", Gear: " + std::to_string(get_gearset())
673:           + ", I_max: " + std::to_string(internal_motor_pid.I_max)
674:           + ", I: " + std::to_string(integral)
675:           + ", IME: " + std::to_string(get_encoder_position())
676:           + ", kD: " + std::to_string(internal_motor_pid.kD)
677:           + ", kI: " + std::to_string(internal_motor_pid.kI)
678:           + ", kP: " + std::to_string(internal_motor_pid.kP)
```

```
679:                 + ", Reversed: " + std::to_string(is_reversed())
680:                 + ", Slew: " + std::to_string(get_slew_rate())
681:                 + ", Target_Vol: " + std::to_string(target_voltage)
682:                 + ", Time: " + std::to_string(pros::millis())
683:                 + ", Vel_Sp: " + std::to_string(calc_target_velocity(target_voltage))
684:                 + ", Vel: " + std::to_string(get_actual_velocity())
685:             );
686:             break;
687:
688:         case 5:
689:             log_msg = (
690:                 "[INFO]" + std::string(" Motor ") + std::to_string(motor_port)
691:                 + ", Actual_Vol: " + std::to_string(get_actual_voltage())
692:                 + ", Brake: " + std::to_string(get_brake_mode())
693:                 + ", Calc_Target_Vol: " + std::to_string(voltage)
694:                 + ", Current: " + std::to_string(get_current_draw())
695:                 + ", Dir: " + std::to_string(get_direction())
696:                 + ", Gear: " + std::to_string(get_gearset())
697:                 + ", I_max: " + std::to_string(internal_motor_pid.I_max)
698:                 + ", I: " + std::to_string(integral)
699:                 + ", IME: " + std::to_string(get_encoder_position())
700:                 + ", kD: " + std::to_string(internal_motor_pid.kD)
701:                 + ", kI: " + std::to_string(internal_motor_pid.kI)
702:                 + ", kP: " + std::to_string(internal_motor_pid.kP)
703:                 + ", Reversed: " + std::to_string(is_reversed())
704:                 + ", Slew: " + std::to_string(get_slew_rate())
705:                 + ", Target_Vol: " + std::to_string(target_voltage)
706:                 + ", Temp: " + std::to_string(get_temperature())
707:                 + ", Time: " + std::to_string(pros::millis())
708:                 + ", Torque: " + std::to_string(get_torque())
709:                 + ", Vel_Sp: " + std::to_string(calc_target_velocity(target_voltage))
710:                 + ", Vel: " + std::to_string(get_actual_velocity())
711:             );
712:             break;
713:
714:     }
715:
716:     std::clog << log_msg << "\n";  //change to actual logger class to avoid fragmentation
717:
718:     return 1;
719: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/motors/MotorThread.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on:
5:   * @reviewed_by:
6:   * TODO:
7:   *
8:   * contains functions that handle motor functions
9:   */
10:
11:  #ifndef __MOTORTHREAD_HPP__
12:  #define __MOTORTHREAD_HPP__
13:
14:  #include <vector>
15:  #include <atomic>
16:
17:  #include "main.h"
18:
19:  #include "../../Configuration.hpp"
20:  #include "Motor.hpp"
21:
22:
23:  // singleton class with thread for running motors
24:  class MotorThread
25:  {
26:      private:
27:          MotorThread();
28:          static MotorThread *thread_obj;
29:
30:          static std::vector<Motor*> motors;
31:          static std::atomic<bool> lock;  //protect vector from concurrent access
32:
33:          static void run(void*);
34:
35:          pros::Task *thread;
36:
37:
38:      public:
39:          ~MotorThread();
40:
41:          /**
42:           * @return: MotorThread -> instance of class to be used throughout program
43:           *
44:           * give user the instance of the singleton class or creates it if it does
45:           * not yet exist
46:           */
47:          static MotorThread* get_instance();
48:
49:          void start_thread();
50:          void stop_thread();
51:
52:          int register_motor( Motor &motor );
53:          int unregister_motor( Motor &motor );
54:
55:
56:  };
57:
58:  #endif
```

```
/**
 * @file: ./RobotCode/src/objects/motors/MotorThread.cpp
 * @author: Aiden Carney
 * @reviewed_on:
 * @reviewed_by:
 * TODO:
 *
 * contains implementation for functions that handle motor functions
 */

#include <vector>
#include <atomic>

#include "main.h"

#include "Motor.hpp"
#include "MotorThread.hpp"


MotorThread *MotorThread::thread_obj = NULL;
std::vector<Motor*> MotorThread::motors;
std::atomic<bool> MotorThread::lock = ATOMIC_VAR_INIT(false);


MotorThread::MotorThread() //:
  //thread( run, (void*)NULL, TASK_PRIORITY_DEFAULT, TASK_STACK_DEPTH_DEFAULT, "motor_thread")
{
  thread = new pros::Task( run, (void*)NULL, TASK_PRIORITY_DEFAULT, TASK_STACK_DEPTH_DEFAULT, "motor_thread");
  thread->suspend();
}


MotorThread::~MotorThread()
{
  thread->remove();
  delete thread;
}


void MotorThread::run(void*)
{
  int start = pros::millis();
  while ( 1 )
  {
    while ( lock.exchange( true ) );
    for ( int i = 0; i < motors.size(); i++ )
    {
      motors.at(i)->run( pros::millis() - start );
    }
    start = pros::millis();
    lock.exchange(false);
    pros::delay(10);
  }
}



/**
 * inits object if object is not already initialized based on a static bool
 * sets bool if it is not set
 */
MotorThread* MotorThread::get_instance()
{
  if ( thread_obj == NULL )
  {
    thread_obj = new MotorThread;
  }
  return thread_obj;
}


void MotorThread::start_thread()
{
  thread->resume();
}

void MotorThread::stop_thread()
{
  thread->suspend();
}


int MotorThread::register_motor( Motor &motor )
{
  while ( lock.exchange( true ) );

  try
  {
    motors.push_back(&motor);
    std::clog << "[INFO] " << pros::millis() << " motor added at " << &motor << "\n";
  }
  catch ( ... )
  {
    std::cerr << "[WARNING] " << pros::millis() << " could not add motor at " << &motor << "\n";
    lock.exchange(false);
    return 0;
  }

  lock.exchange(false);
  return 1;
}


int MotorThread::unregister_motor( Motor &motor )
{
  while ( lock.exchange( true ) );

  auto element = std::find(begin(motors), end(motors), &motor);
  if ( element != motors.end())
  {
    motors.erase(element);
    std::clog << "[INFO] " << pros::millis() << " motor removed at " << &motor << "\n";
  }
```

```
114:     else
115:     {
116:         std::cerr << "[WARNING] " << pros::millis() << " could not remove motor at " << &motor << "\n";
117:         return 0;
118:     }
119:
120:     lock.exchange(false);
121:     return 1;
122:
123: }
```

```
  1: /**
  2:  * @file: ./RobotCode/src/motors/Motors.hpp
  3:  * @author: Aiden Carney
  4:  * @reviewed_on: 11/8/19
  5:  * @reviewed_by: Aiden Carney
  6:  * TODO: add function for updating motors
  7:  *
  8:  * contains singleton class for all the motors
  9:  */
 10:
 11: #ifndef __MOTORS_HPP__
 12: #define __MOTORS_HPP__
 13:
 14: #include <string>
 15: #include <unordered_map>
 16:
 17: #include "../../../include/api.h"
 18: #include "../../../include/main.h"
 19: #include "../../../include/pros/motors.hpp"
 20: #include "../../../include/pros/rtos.hpp"
 21:
 22: #include "../../Configuration.hpp"
 23:
 24: //motor port definitions
 25: #define FRONTRIGHT_PORT    5
 26: #define BACKRIGHT_PORT     2
 27: #define FRONTLEFT_PORT     3
 28: #define BACKLEFT_PORT      4
 29: #define LEFT_INTAKE_PORT   6
 30: #define RIGHT_INTAKE_PORT  8
 31: #define TILTER_PORT        7
 32: #define LIFT_PORT          9
 33:
 34:
 35:
 36: /**
 37:  * @see: ../../Configuration.hpp
 38:  *
 39:  * has motor functions to be used throughout program
 40:  * exists so that all motors are in one place and can be changed easily
 41:  * purpose of class line 3
 42:  */
 43: class Motors
 44: {
 45:     private:
 46:         Motors();
 47:         static Motors *motors_obj;
 48:
 49:         static std::string bin_string( pros::Motor* motor );
 50:
 51:
 52:     public:
 53:         ~Motors();
 54:
 55:         /**
 56:          * @return: Motors* -> instance of class to be used throughout program
 57:          *
 58:          * give user the instance of the singleton class or creates it if it does
 59:          * not yet exist
 60:          */
 61:         static Motors* get_instance();
 62:
 63:
 64:         bool allow_left_chassis; //bools allow for autonomous commands
 65:                                  //to be run in opcontrol
 66:                                  //eg. motor defaults to off
 67:                                  //in opcontrol, so if task is being
 68:                                  //run motor will stall
 69:         bool allow_right_chassis;
 70:         bool allow_intake;
 71:         bool allow_tilter;
 72:         bool allow_lift;
 73:
 74:         pros::Motor *frontRight; //drive motors
 75:         pros::Motor *backRight;
 76:         pros::Motor *frontLeft;
 77:         pros::Motor *backLeft;
 78:
 79:         pros::Motor *right_intake; //intake motors
 80:         pros::Motor *left_intake;
 81:
 82:         pros::Motor *tilter; //tilter motor
 83:
 84:         pros::Motor *lift; //other motor
 85:
 86:         /**
 87:          * @return: None
 88:          *
 89:          * @see: ../writer/Writer.hpp
 90:          *
 91:          * takes information from motor at 1 second intervals and sends
 92:          * it to the writer queue
 93:          */
 94:         void record_macro();
 95:
 96:
 97:         /**
 98:          * @return: None
 99:          *
100:          * TODO: add actual content to function
101:          *
102:          * takes data from the sd card for motor movements over an interval of time
103:          * with the intent of repeating the action that occured at that time
104:          */
105:         void run_macro();
106:
107:
108: };
109:
110:
111:
112:
113: #endif
```

```
1:  #include <algorithm>
2:  #include <array>
3:  #include <bitset>
4:  #include <chrono>
5:  #include <ctime>
6:  #include <iostream>
7:  #include <string>
8:  #include <unordered_map>
9:  #include <utility>
10: #include <vector>
11: #include <errno.h>
12: #include <cerrno>
13: #include <cstring>
14: #include <clocale>
15: #include <fstream>
16:
17: #include "../../../include/api.h"
18: #include "../../../include/main.h"
19: #include "../../../include/pros/motors.hpp"
20: #include "../../../include/pros/rtos.hpp"
21:
22: #include "../../../lib/date/date.h"
23: #include "../../Configuration.hpp"
24: #include "../writer/Writer.hpp"
25: #include "Motors.hpp"
26:
27:
28:
29:
30: #ifdef MACRO_RECORD_DEBUG
31:    std::array< std::array<int, 1001>, 5 >numbers;
32: #endif
33:
34: Motors *Motors::motors_obj = NULL;
35:
36:
37:
38: //default instance constructor
39: Motors::Motors()
40: {
41:    allow_left_chassis = true;
42:    allow_right_chassis = true;
43:    allow_intake = true;
44:    allow_tilter = true;
45:    allow_lift = true;
46:
47:    //drive
48:    frontRight = new pros::Motor(Configuration::get_instance()->front_right_port, pros::E_MOTOR_GEARSET_18, Configuration::get_instance()->front_right_reversed, pros::E_MOTOR_ENCODER_DEGREES);
49:    backRight = new pros::Motor(Configuration::get_instance()->back_left_port, pros::E_MOTOR_GEARSET_18, Configuration::get_instance()->back_right_reversed, pros::E_MOTOR_ENCODER_DEGREES);
50:    frontLeft = new pros::Motor(Configuration::get_instance()->front_left_port, pros::E_MOTOR_GEARSET_18, 0, pros::E_MOTOR_ENCODER_DEGREES);
51:    backLeft = new pros::Motor(Configuration::get_instance()->back_right_port, pros::E_MOTOR_GEARSET_18, 0, pros::E_MOTOR_ENCODER_DEGREES);
52:
53:    //intake
54:    left_intake = new pros::Motor(Configuration::get_instance()->left_intake_port, pros::E_MOTOR_GEARSET_18, Configuration::get_instance()->left_intake_reversed, pros::E_MOTOR_ENCODER_DEGREES);
55:    right_intake = new pros::Motor(Configuration::get_instance()->right_intake_port, pros::E_MOTOR_GEARSET_18, Configuration::get_instance()->right_intake_reversed, pros::E_MOTOR_ENCODER_DEGREES);
56:
57:    //tilter
58:    tilter = new pros::Motor(Configuration::get_instance()->tilter_port, pros::E_MOTOR_GEARSET_18, Configuration::get_instance()->tilter_reversed, pros::E_MOTOR_ENCODER_DEGREES);
59:
60:    //lift
61:    lift = new pros::Motor(Configuration::get_instance()->lift_port, pros::E_MOTOR_GEARSET_18, Configuration::get_instance()->lift_reversed, pros::E_MOTOR_ENCODER_DEGREES);
62: }
63:
64:
65: //instance destructor
66: Motors::~Motors()
67: {
68:
69: }
70:
71:
72:
73: /**
74:  * inits object if object is not already initialized based on a static bool
75:  * sets bool if it is not set
76:  */
77: Motors* Motors::get_instance()
78: {
79:    if ( motors_obj == NULL )
80:    {
81:       motors_obj = new Motors;
82:    }
83:    return motors_obj;
84: }
85:
86:
87:
88:
89: /**
90:  * returns information about a motor in a string of bits
91:  * this string is always 12 bits long
92:  */
93: std::string Motors::bin_string( pros::Motor* motor )
94: {
95:    std::string str;
96:    str += motor->get_current_draw() ? "1" : "0";
97:    str += ( motor->get_direction() > 0 ) ? "1" : "0";
98:    str += std::bitset<10>( (int)std::abs( motor->get_actual_velocity() ) ).to_string();
99:
100:   return str;
101: }
102:
103:
104:
105: /**
106:  * starts recording the macro, if MACRO_RECORD_DEBUG is set, then data will also
107:  * be written that can be graphed to see if the program is working
108:  */
109: void Motors::record_macro()
110: {
111:    #ifdef MACRO_RECORD_DEBUG
112:       int n = 0;
113:       for ( int i = 0; i < 5; i++ )
```

```
114:        {
115:           for ( int j = 0; j < 1001; j++ )
116:           {
117:              numbers[i][j] = n;
118:              n++;
119:           }
120:        }
121:     #endif
122:
123:
124:     //std::vector< std::pair< pros::Motor, std::string> > *test1 = new std::vector< std::pair< pros::Motor, std::string> >;
125:
126:     std::vector< std::pair< pros::Motor*, std::string> > motors;
127:
128:     motors.push_back( std::make_pair( frontRight, "thread1" ) );
129:     motors.push_back( std::make_pair( frontRight, "thread2" ) );
130:     motors.push_back( std::make_pair( frontRight, "thread3" ) );
131:     motors.push_back( std::make_pair( frontRight, "thread4" ) );
132:     motors.push_back( std::make_pair( frontRight, "thread5" ) );
133:     //test->push_back( std::make_pair( frontRight, "frontRight6" ) );
134:     //test->push_back( std::make_pair( frontRight, "frontRight7" ) );
135:     //test->push_back( std::make_pair( frontRight, "frontRight8" ) );
136:
137:  /*    pros::Task macro_test_task1 (record,
138:                              (void*)test1,
139:                              TASK_PRIORITY_DEFAULT,
140:                              TASK_STACK_DEPTH_DEFAULT,
141:                              "macro_test_task1" );*/
142:
143:
144:     Writer writer;
145:     //td::vector< std::pair< pros::Motor, std::string> > *m = static_cast <std::vector< std::pair< pros::Motor, std::string> > *> (test1);
146:     //std::vector< std::pair< pros::Motor, std::string> > motors = *m;
147:
148:     while ( pros::millis() < 30000 )
149:     {
150:        //std::cout << writer.get_num_files_open() << "\n";
151:        //run unit test and tell each thread to send an item to the queue
152:        //every x milliseconds
153:        auto start = std::chrono::high_resolution_clock::now();
154:
155:        //macro_test_task1.notify();
156:        for ( int i = 0; i < motors.size(); i++ )
157:        {
158:           #ifdef MACRO_RECORD_DEBUG
159:              std::string str = (
160:                 std::to_string( numbers[i][index] )
161:                 + ","
162:                 + date::format("%d-%m-%Y %T", date::floor<std::chrono::microseconds>(std::chrono::system_clock::now()))
163:                 + ","
164:                 + date::format("%d-%m-%Y %T", date::floor<std::chrono::microseconds>(std::chrono::system_clock::now()))
165:                 + "\n"
166:              );
167:              std::string file_name = std::string("/usd/log/15/") + std::get<1>( motors[i] ) + "/data.csv";
168:           #else
169:              std::string str = bin_string( std::get<0>( motors[i] ) );
170:              std::string file_name = std::string("/usd/macros/") + std::get<1>( motors[i] ) + "_temp";
171:           #endif
172:
173:           writer_obj w = { file_name, "a", str };
174:           writer.add( w );
175:        }
176:        //std::cout << pros::millis() << " " << writer.get_count() << "\n";
177:        auto stop = std::chrono::high_resolution_clock::now();
178:        auto duration = std::chrono::duration_cast<std::chrono::microseconds>(stop - start);
179:        int delta_t = ( duration.count() <= 1000 ) ? duration.count() : 1000;  //counted in microseconds, limited to 1000 microseconds
180:        int wait_time = ( 1000 - delta_t ) / 1000;  //convert to milliseconds because
181:                                                    //that is what the system clock
182:                                                    //uses
183:        pros::delay(wait_time);
184:
185:        //std::cout << writer.get_count() << " " << pros::millis() << " " << wait_time << " " << duration.count() << " " << delta_t << "\n";
186:        //writer.dump();
187:     }
188:     std::cout << "task ended\n";
189:     //std::cout << "done waiting\n";
190:     //macro_test_task1.remove();
191:
192:     //Check for Completion
193:     // write eof at end of file
194:     //wait until each file has "eof" at end of file so that it is known that it
195:     //is finished being written to
196:     #ifdef MACRO_RECORD_DEBUG
197:
198:  /*       Writer writer;
199:        std::vector< std::pair< pros::Motor, std::string> > *m = static_cast <std::vector< std::pair< pros::Motor, std::string> > *> (test1);
200:        std::vector< std::pair< pros::Motor, std::string> > motors = *m;*/
201:
202:        //std::cout << "writing eof\n";
203:        //add eof so that macro can be recorded
204:        for ( int i = 0; i < motors.size(); i++ )
205:        {
206:           //std::string file_name = std::string("/usd/log/15/") + std::get<1>( motors[i] ) + "/data.csv";
207:           std::string file_name = std::string("/usd/macros/") + std::get<1>( motors[i] );
208:           writer_obj w = { file_name, "a", "eof" };
209:           writer.add( w );
210:        }
211:
212:        //std::cout << "checking for eof to be written\n";
213:        //wait for eof to be written
214:        for( int i = 0; i < motors.size(); i++ )
215:        {
216:           std::string file_name = std::string("/usd/macros/") + std::get<1>( motors[i] ) + "_temp";
217:           std::string last_char;
218:           while( last_char.compare("f") == 0 ) //check if last char is f from "eof" -> signifies file is finished
219:                                                //being written to
220:           {
221:              std::ifstream fileIn( file_name );  //Read file
222:
223:              std::stringstream buffer;
224:              buffer << fileIn.rdbuf();
225:              last_char = buffer.str().back();
226:
```

```
227:                fileIn.close();
228:                pros::delay(50);  //don't starve processer in a passive area
229:            }
230:
231:            writer.rename( file_name, std::string("/usd/macros/") + std::get<1>( motors[i] ));
232:            writer.remove( file_name );
233:        }
234:        //std::cout << "function exiting\n";
235:
236:    #endif
237:
238:
239:
240: }
241:
242:
243:
244: /**
245:  * does nothing at the moment
246:  */
247: void Motors::run_macro()
248: {
249:
250: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/sensors/Sensors.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/4/19
5:   * @reviewed_by: Aiden Carney
6:   * TODO: move defines to configuration file as well as make a singleton class
7:   *
8:   * contains a class for interacting with the ADI sensors on the robot
9:   */
10:
11: #ifndef __SENSORS_HPP__
12: #define __SENSORS_HPP__
13:
14: #include "../../../include/main.h"
15: #include "../../../include/api.h"
16: #include "../../../include/pros/rtos.hpp"
17: #include "../../../include/pros/motors.hpp"
18:
19:
20: //sensor port definitions
21: #define GYRO1_PORT          'A'
22: #define GYRO2_PORT          'B'
23: #define ACCELLEROMETERX_PORT  'C'
24: #define ACCELLEROMETERY_PORT  'D'
25: #define ACCELLEROMETERZ_PORT  'E'
26: #define POTENTIOMETER_PORT    'F'
27: #define LIMITSWITCH_PORT      'G'
28: #define LED_PORT              'H'
29: #define VISIONSENSOR_PORT     20
30:
31:
32: //structs used to return sensor readings from functions
33: struct gyroValues
34: {
35:     float gyro1 = 0;
36:     float gyro2 = 0;
37: };
38:
39: struct accelValues
40: {
41:     float Xaxis = 0;
42:     float Yaxis = 0;
43:     float Zaxis = 0;
44: };
45:
46:
47:
48: /**
49:  * @see: pros docs
50:  *
51:  * contains methods for accessing ADI sensor values
52:  */
53: class Sensors
54: {
55:     private:
56:
57:
58:     public:
59:         Sensors();
60:         Sensors(bool calibrate);
61:         ~Sensors();
62:
63:         static const pros::ADIAnalogIn chassisGyro1;
64:         static const pros::ADIAnalogIn chassisGyro2;
65:         static const pros::ADIAnalogIn accellerometerX;
66:         static const pros::ADIAnalogIn accellerometerY;
67:         static const pros::ADIAnalogIn accellerometerZ;
68:         static const pros::ADIAnalogIn potentiometer;
69:         static const pros::ADIDigitalIn limitSwitch;
70:         static const pros::ADIDigitalOut led;
71:         static const pros::Vision vision_sensor;
72:
73:         //for other areas of code to know if analog sensors have
74:         //been calibrated
75:         static bool gyroCalibrated;
76:         static bool accelCalibrated;
77:         static bool potCalibrated;
78:
79:         //calibrates analog sensors
80:
81:         /**
82:          * @return: None
83:          *
84:          * @see: pros docs
85:          *
86:          * calibrates two gyros by making a call to the pros api
87:          * this is a blocking function
88:          * calibrated flag for the sensor is set
89:          */
90:         static void calibrateGyro();
91:
92:         /**
93:          * @return: None
94:          *
95:          * @see: pros docs
96:          *
97:          * calibrates acceleraometer by making a call to the pros api
98:          * this is a blocking function
99:          * calibrated flag for the sensor is set
100:         */
101:        static void calibrateAccel();
102:
103:        /**
104:         * @return: None
105:         *
106:         * @see: pros docs
107:         *
108:         * calibrates the potentiometer by making a call to the pros api
109:         * this is a blocking function
110:         * calibrated flag for the sensor is set
111:         */
112:        static void calibratePot();
113:
```

```
114:        static bool led_status;
115:
116:        /**
117:         * @return: gyroValues -> struct that contains the current raw gyro readings
118:         *
119:         * @see: pros docs
120:         *
121:         * returns the uncorrected gyro reading
122:         * if the gyro has not been calibrated, return INT32_MAX
123:         */
124:        gyroValues getRawGyro();
125:
126:        /**
127:         * @return: gyroValues -> struct that contains the current corrected gyro readings
128:         *
129:         * @see: pros docs
130:         *
131:         * returns the calibrated/corrected gyro reading
132:         * if the gyro has not been calibrated, return INT32_MAX
133:         */
134:        gyroValues getCorrectedGyro();
135:
136:
137:
138:
139:        /**
140:         * @return: accelValues -> struct that contains the current raw accelerometer readings
141:         *
142:         * @see: pros docs
143:         *
144:         * returns the uncorrected accelerometer reading
145:         * if the accelerometer has not been calibrated, return INT32_MAX
146:         */
147:        accelValues getRawAccelerometer();
148:
149:        /**
150:         * @return: accelValues -> struct that contains the current corrected accelerometer readings
151:         *
152:         * @see: pros docs
153:         *
154:         * returns the corrected accelerometer reading
155:         * if the accelerometer has not been calibrated, return INT32_MAX
156:         */
157:        accelValues getCorrectedAccelerometer();
158:
159:
160:
161:
162:        /**
163:         * @return: float -> current raw potentiometer readings
164:         *
165:         * @see: pros docs
166:         *
167:         * returns the uncorrected potentiometer reading
168:         * if the potentiometer has not been calibrated, return INT32_MAX
169:         */
170:        float getRawPot();
171:
172:        /**
173:         * @return: float -> current corrected potentiometer readings
174:         *
175:         * @see: pros docs
176:         *
177:         * returns the corrected potentiometer reading
178:         * if the potentiometer has not been calibrated, return INT32_MAX
179:         */
180:        float getCorrectedPot();
181:
182:
183:
184:
185:        /**
186:         * @return: bool -> current value of the limit switch
187:         *
188:         * @see: pros docs
189:         *
190:         * returns true if the limit switch is touched and false if otherwise
191:         */
192:        bool getLimitSwitch();
193:
194:
195:
196:
197:        /**
198:         * @return: None
199:         *
200:         * @see: pros docs
201:         *
202:         * sets value of led to on
203:         */
204:        void set_led();
205:
206:        /**
207:         * @return: None
208:         *
209:         * @see: pros docs
210:         *
211:         * sets value of led to off
212:         */
213:        void clear_led();
214:
215:    };
216:
217:
218:
219:
220:    #endif
```

```
/**
 * @file: ./RobotCode/src/objects/sensors/Sensors.cpp
 * @author: Aiden Carney
 * @reviewed_on: 12/4/19
 * @reviewed_by: Aiden Carney
 *
 * @see: Sensors.hpp
 *
 * contains definitions for sensors and implementation for sensor class
 */

#include "../../../include/main.h"
#include "../../../include/api.h"
#include "../../../include/pros/rtos.hpp"
#include "../../../include/pros/motors.hpp"

#include "Sensors.hpp"

bool Sensors::led_status = false;

bool Sensors::gyroCalibrated = false;
bool Sensors::accelCalibrated = false;
bool Sensors::potCalibrated = false;

//do not initialize gyros ports or calibrate accelerometer
//because they will block for 1 sec to calibrate
//init of gyros and accelerometer occurs in seperate functions
const pros::ADIAnalogIn Sensors::chassisGyro1(GYRO1_PORT);
const pros::ADIAnalogIn Sensors::chassisGyro2(GYRO2_PORT);
const pros::ADIAnalogIn Sensors::accellerometerX(ACCELLEROMETERX_PORT);
const pros::ADIAnalogIn Sensors::accellerometerY(ACCELLEROMETERY_PORT);
const pros::ADIAnalogIn Sensors::accellerometerZ(ACCELLEROMETERZ_PORT);
const pros::ADIAnalogIn Sensors::potentiometer(POTENTIOMETER_PORT);
const pros::ADIDigitalIn Sensors::limitSwitch(LIMITSWITCH_PORT);
const pros::ADIDigitalOut Sensors::led(LED_PORT);
const pros::Vision Sensors::vision_sensor(VISIONSENSOR_PORT);


Sensors::Sensors()
{
    //does not calibrate analog sensors because they will block
    //and this may be called at unwanted times
}



Sensors::~Sensors()
{

}




/**
 * calibrates gyros
 * will block for approximately 2 sec
 * sets gyro calibrated flag
 */
void Sensors::calibrateGyro()
                        //calibrate must be explicitly called in
                        //order for gyro to be calibrated
                        //this is with the intent to save time having
                        //to keep the robot still to wait for
                        //calibration to finish or risk it failing
{
    //init gyros, must be called before
    //any gyro functions are called
    //gyros will be calibrated either by
    //debugger or in initialize before
    //auton
    chassisGyro1.calibrate();
    chassisGyro2.calibrate();

    gyroCalibrated = true; //mainly for lcd to warn user to keep robot still
                    //if gyros have not been calibrated yet


}



/**
 * calibrates accelerometer
 * blocks for approximately 1 sec
 * sets accelerometer calibrated flag
 */
void Sensors::calibrateAccel()
                    //calibrate must be explicitly called in
                    //order for accelerometer to be calibrated
                    //this is with the intent to save time having
                    //to keep the robot still to wait for
                    //calibration to finish or risk it failing
{

    //calibrate accelerometer, must be called before
    //any accelerometer functions are called
    //accelerometer will be calibrated either by
    //debugger or in initialize before
    //auton
    accellerometerX.calibrate();
    accellerometerY.calibrate();
    accellerometerZ.calibrate();

    accelCalibrated = true; //mainly for lcd to warn user to keep robot still

}
```

```
114:
115: /**
116:  * calibrates potentiometer
117:  * blocks for approximately 1 sec
118:  * sets potentiometer calibrated flag
119:  */
120: void Sensors::calibratePot()
121:                     //calibrate must be explicitly called in
122:                     //order for potentiometer to be calibrated
123:                     //this is with the intent to save time having
124:                     //to keep the robot still to wait for
125:                     //calibration to finish or risk it failing
126: {
127:   //calibrate potentiometer, must be called before
128:   //any potentiometer functions are called
129:   //potentiometer will be calibrated either by
130:   //debugger or in initialize before
131:   //auton
132:   potentiometer.calibrate();
133:
134:   potCalibrated = true; //mainly for lcd to warn user to keep robot still
135: }
136:
137:
138:
139:
140: /**
141:  * makes a gyro values struct and adds the uncorrected values to it
142:  * if gyro has not been calibrated, values are set to INT32_MAX
143:  */
144: gyroValues Sensors::getRawGyro()
145: {
146:   gyroValues gyro_vals;
147:   if (gyroCalibrated)
148:   {
149:     gyro_vals.gyro1 = chassisGyro1.get_value();
150:     gyro_vals.gyro2 = chassisGyro2.get_value();
151:   }
152:   else
153:   {
154:     gyro_vals.gyro1 = INT32_MAX;
155:     gyro_vals.gyro2 = INT32_MAX;
156:   }
157:
158:   return gyro_vals;
159: }
160:
161:
162:
163:
164: /**
165:  * makes a gyro values struct and adds the calibrated values to it
166:  * if gyro has not been calibrated, values are set to INT32_MAX
167:  */
168: gyroValues Sensors::getCorrectedGyro()
169: {
170:   gyroValues gyro_vals;
171:   if (gyroCalibrated)
172:   {
173:     gyro_vals.gyro1 = chassisGyro1.get_value_calibrated();
174:     gyro_vals.gyro2 = chassisGyro2.get_value_calibrated();
175:   }
176:   else
177:   {
178:     gyro_vals.gyro1 = INT32_MAX;
179:     gyro_vals.gyro2 = INT32_MAX;
180:   }
181:
182:   return gyro_vals;
183: }
184:
185:
186:
187:
188: /**
189:  * makes a accelerometer values struct and adds the uncorrected values to it
190:  * if accelerometer has not been calibrated, values are set to INT32_MAX
191:  */
192: accelValues Sensors::getRawAccelerometer()
193: {
194:   accelValues accel_vals;
195:   if (accelCalibrated)
196:   {
197:     accel_vals.Xaxis = accellerometerX.get_value();
198:     accel_vals.Yaxis = accellerometerY.get_value();
199:     accel_vals.Zaxis = accellerometerZ.get_value();
200:   }
201:   else
202:   {
203:     accel_vals.Xaxis = INT32_MAX;
204:     accel_vals.Yaxis = INT32_MAX;
205:     accel_vals.Zaxis = INT32_MAX;
206:   }
207:
208:   return accel_vals;
209: }
210:
211:
212:
213:
214: /**
215:  * makes a accelerometer values struct and adds the calibrated values to it
216:  * if accelerometer has not been calibrated, values are set to INT32_MAX
217:  */
218: accelValues Sensors::getCorrectedAccelerometer()
219: {
220:   accelValues accel_vals;
221:   if (accelCalibrated)
222:   {
223:     accel_vals.Xaxis = accellerometerX.get_value_calibrated();
224:     accel_vals.Yaxis = accellerometerY.get_value_calibrated();
225:     accel_vals.Zaxis = accellerometerZ.get_value_calibrated();
226:   }
```

```
227:     else
228:     {
229:         accel_vals.Xaxis = INT32_MAX;
230:         accel_vals.Yaxis = INT32_MAX;
231:         accel_vals.Zaxis = INT32_MAX;
232:     }
233:
234:     return accel_vals;
235: }
236:
237:
238:
239:
240: /**
241:  * makes a potentiometer values struct and adds the uncorrected values to it
242:  * if potentiometer has not been calibrated, values are set to INT32_MAX
243:  */
244: float Sensors::getRawPot()
245: {
246:     float val;
247:     if (potCalibrated)
248:     {
249:         val = potentiometer.get_value();
250:     }
251:
252:     else
253:     {
254:         val = INT32_MAX;
255:     }
256:
257:     return val;
258: }
259:
260:
261:
262:
263: /**
264:  * makes a potentiometer values struct and adds the corrected values to it
265:  * if potentiometer has not been calibrated, values are set to INT32_MAX
266:  */
267: float Sensors::getCorrectedPot()
268: {
269:     float val;
270:     if (potCalibrated)
271:     {
272:         val = potentiometer.get_value_calibrated();
273:     }
274:
275:     else
276:     {
277:         val = INT32_MAX;
278:     }
279:
280:     return val;
281: }
282:
283:
284:
285:
286: /**
287:  * gets the value of the limit switch as either true or false
288:  * based on if the switch is being touched
289:  */
290: bool Sensors::getLimitSwitch()
291: {
292:     return limitSwitch.get_value();
293: }
294:
295:
296:
297:
298: /**
299:  * turns led on and sets the led status flag
300:  */
301: void Sensors::set_led()
302: {
303:     led.set_value(true);
304:     led_status = true;
305: }
306:
307:
308:
309:
310: /**
311:  * turns led off and clears led status flag
312:  */
313: void Sensors::clear_led()
314: {
315:     led.set_value(false);
316:     led_status = false;
317: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/robotChassis/chassis.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/4/19
5:   * @reviewed_by: Aiden Carney
6:   * TODO: Clean up includes
7:   *
8:   * Contains class for the chassis subsystem
9:   * has methods for driving during autonomous including turning and driving straight
10:  */
11:
12: #ifndef __CHASSIS_HPP__
13: #define __CHASSIS_HPP__
14:
15: #include "../../../include/main.h"
16: #include "../../../include/api.h"
17: #include "../../../include/pros/rtos.hpp"
18: #include "../../../include/pros/motors.hpp"
19:
20: #include "../motors/Motors.hpp"
21: #include "../sensors/Sensors.hpp"
22:
23:
24: /**
25:  * @see: Motors.hpp
26:  *
27:  * contains methods to allow for easy control of the robot during
28:  * the autonomous period
29:  */
30: class Chassis :
31:     public Sensors
32: {
33:     private:
34:         bool reversed;
35:         bool motorSlew;
36:         int slewMaxSpeed;
37:         int numMotors;
38:
39:
40:     public:
41:         Chassis();
42:         ~Chassis();
43:
44:         /**
45:          * @param: int revolutions -> how the setpoint in encoder ticks for the robot position
46:          * @param: int velocity -> percentage of how fast the robot will move
47:          * @return: None
48:          *
49:          * @see: Motors.hpp
50:          *
51:          * TODO: add custom PID distance control, acceleration control, and straight drive
52:          *
53:          * allows the robot to move straight towards a setpoint
54:          */
55:         void straight(int revolutions, int velocity, int timeout=INT32_MAX); //drive functions
56:
57:
58:         /**
59:          * @param: int revolutions -> the setpoint for the motor encoders
60:          * @return: None
61:          *
62:          * @see: Motors.hpp
63:          *
64:          *
65:          * TODO: add velocity parameter, custom PID controller
66:          *
67:          * turns left at full speed by turning both wheels of the robot
68:          * in opposite directions
69:          */
70:         void turnLeft(int revolutions, int timeout=INT32_MAX); //turn functions
71:
72:         /**
73:          * @param: int revolutions -> the setpoint for the motor encoders
74:          * @return: None
75:          *
76:          * @see: Motors.hpp
77:          *
78:          *
79:          * TODO: add velocity parameter, custom PID controller
80:          *
81:          * turns right at full speed by turning both wheels of the robot
82:          * in opposite directions
83:          */
84:         void turnRight(int revolutions, int timeout=INT32_MAX);
85:
86:         /**
87:          * @param: int revolutions -> the setpoint for the motor encoders
88:          * @return: None
89:          *
90:          * @see: Motors.hpp
91:          *
92:          * TODO: add functionality
93:          *
94:          * does nothing
95:          */
96:         void leftSide(int revolutions);
97:
98:         /**
99:          * @param: int revolutions -> the setpoint for the motor encoders
100:         * @return: None
101:         *
102:         * @see: Motors.hpp
103:         *
104:         * TODO: add functionality
105:         *
106:         * does nothing
107:         */
108:        void rightSide(int revolutions);
109:
110:
111:
112:
113:        /**
```

```
114:         * @return: None
115:         *
116:         * @see: Motors.hpp
117:         *
118:         * changes the direction at the api motor level so that all the
119:         * motors in the chassis system are reversed
120:         * useful for allowing to change direction of drive in user control
121:         */
122:        void changeDirection(); //misc functions
123:
124:        /**
125:         * @param: int speed -> the new speed the slew rate controller
126:         * @return: None
127:         *
128:         * TODO: add data validation
129:         *
130:         * Setter function for private variable int slewMaxSpeed
131:         */
132:        void setSlewMaxSpeed(int speed);
133:    };
134:
135:
136:
137:
138:    #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/robotChassis/chassis.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/4/19
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: chassis.hpp
8:   *
9:   * contains implementation for chassis subsytem class
10:  */
11:
12:  #include "../../../include/main.h"
13:  #include "../../../include/api.h"
14:  #include "../../../include/pros/rtos.hpp"
15:  #include "../../../include/pros/motors.hpp"
16:
17:  #include "chassis.hpp"
18:
19:
20:
21:  Chassis::Chassis()
22:  {
23:    Motors *motors = Motors::get_instance();
24:
25:    reversed = false;
26:    motorSlew = false;
27:    slewMaxSpeed = 100;
28:
29:    numMotors = 4;
30:
31:    motors->frontRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
32:    motors->backRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
33:    motors->frontLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
34:    motors->backLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
35:  }
36:
37:
38:  Chassis::~Chassis()
39:  {
40:
41:  }
42:
43:
44:
45:
46:
47:  //other methods
48:
49:  /**
50:   * sets motors to spin to a setpoint at a given velocity
51:   * waits until motors are in range of 10 encoder units before stopping
52:   * the motors
53:   */
54:  void Chassis::straight(int revolutions, int velocity, int timeout /*INT32_MAX*/ ) //drives straight
55:  {
56:    Motors *motors = Motors::get_instance();
57:
58:    motors->frontRight->tare_position();
59:    motors->frontLeft->tare_position();
60:    motors->backRight->tare_position();
61:    motors->backLeft->tare_position();
62:
63:    motors->frontRight->move_absolute(revolutions, velocity);
64:    motors->frontLeft->move_absolute(revolutions, velocity);
65:    motors->backRight->move_absolute(revolutions, velocity);
66:    motors->backLeft->move_absolute(revolutions, velocity);
67:
68:    int start = pros::millis();
69:    int time_elapsed = pros::millis() - start;
70:
71:    while ( ( std::abs(revolutions - motors->frontRight->get_position()) ) > 10
72:       && std::abs(revolutions - motors->frontLeft->get_position()) > 10
73:       && time_elapsed < timeout
74:    )
75:    {
76:       pros::delay(10);
77:       time_elapsed = pros::millis() - start;
78:    }
79:
80:    motors->frontRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
81:    motors->backRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
82:    motors->frontLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
83:    motors->backLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
84:
85:  }
86:
87:
88:
89:
90:  /**
91:   * sets right motors to move forwards and left motors to move backwards
92:   * until they hit a given setpoint
93:   * stops motors once they are within 10 endoder units of the setpoint
94:   */
95:  void Chassis::turnLeft(int revolutions, int timeout /*INT32_MAX*/ ) //turns left with wheels on both
96:                                  //sides of chassis spinning
97:  {
98:    Motors *motors = Motors::get_instance();
99:
100:   motors->frontRight->tare_position();
101:   motors->frontLeft->tare_position();
102:   motors->backRight->tare_position();
103:   motors->backLeft->tare_position();
104:
105:   motors->frontRight->move_absolute(revolutions, 100);
106:   motors->frontLeft->move_absolute(0-revolutions, 100);
107:   motors->backRight->move_absolute(revolutions, 100);
108:   motors->backLeft->move_absolute(0-revolutions, 100);
109:
110:   while ( ( std::abs(revolutions - motors->frontRight->get_position()) ) > 10 && std::abs(revolutions - motors->frontLeft->get_position()) > 10)
111:   {
112:      pros::delay(10);
113:   }
```

```
114:
115:     motors->frontRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
116:     motors->backRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
117:     motors->frontLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
118:     motors->backLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
119:   }
120:
121:
122:
123:
124:   /**
125:    * sets left motors to move forwards and right motors to move backwards
126:    * until they hit a given setpoint
127:    * stops motors once they are within 10 endoder units of the setpoint
128:    */
129:   void Chassis::turnRight(int revolutions, int timeout /*INT32_MAX*/ ) //turns right with wheels on both
130:                           //sides of chassis spinning
131:   {
132:     Motors *motors = Motors::get_instance();
133:
134:     motors->frontRight->tare_position();
135:     motors->frontLeft->tare_position();
136:     motors->backRight->tare_position();
137:     motors->backLeft->tare_position();
138:
139:     motors->frontRight->move_absolute(0-revolutions, 100);
140:     motors->frontLeft->move_absolute(revolutions, 100);
141:     motors->backRight->move_absolute(0-revolutions, 100);
142:     motors->backLeft->move_absolute(revolutions, 100);
143:
144:     while ( ( std::abs(revolutions - motors->frontRight->get_position()) ) > 10 && std::abs(revolutions - motors->frontLeft->get_position()) > 10)
145:     {
146:       pros::delay(10);
147:     }
148:
149:     motors->frontRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
150:     motors->backRight->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
151:     motors->frontLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
152:     motors->backLeft->set_brake_mode(pros::E_MOTOR_BRAKE_COAST);
153:   }
154:
155:
156:
157:
158:   /**
159:    * does nothing
160:    */
161:   void Chassis::leftSide(int revolutions) //starts left side of robot
162:                          //used in auton
163:   {
164:
165:   }
166:
167:
168:
169:
170:   /**
171:    * does nothing
172:    */
173:   void Chassis::rightSide(int revolutions) //starts right side of robot
174:                          //used in auton
175:   {
176:
177:   }
178:
179:
180:
181:
182:   /**
183:    * sets all chassis motors to the opposite direction that they were facing
184:    * ie. reversed is now normal and normal is now reversed
185:    */
186:   void Chassis::changeDirection() //changes orientation of robot
187:   {
188:     Motors *motors = Motors::get_instance();
189:
190:     reversed = !(reversed);
191:
192:     motors->frontRight->set_reversed(!(motors->frontRight->is_reversed()));
193:     motors->frontLeft->set_reversed(!(motors->frontLeft->is_reversed()));
194:     motors->backLeft->set_reversed(!(motors->backLeft->is_reversed()));
195:     motors->backRight->set_reversed(!(motors->backRight->is_reversed()));
196:   }
197:
198:
199:
200:
201:   /**
202:    * sets the private variable to the parameter given
203:    */
204:   void Chassis::setSlewMaxSpeed(int speed) //sets max velocity used by
205:                                 //acceleration control
206:   {
207:     slewMaxSpeed = speed;
208:   }
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/lift/Lift.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/5/19
5:   * @reviewed_by: Aiden Carney
6:   * TODO: Test implementation with robot especially with setpoint and move_to
7:   *
8:   * contains class for tilter sub system and commands to move it
9:   */
10:
11: #ifndef __LIFT_HPP__
12: #define __LIFT_HPP__
13:
14: #include <array>
15:
16: #include "../../../include/main.h"
17: #include "../../../include/pros/motors.hpp"
18:
19: #include "../motors/Motors.hpp"
20: #include "../sensors/Sensors.hpp"
21:
22:
23:
24:
25: /**
26:  * @see: Sensors.hpp
27:  * @see: Motors.hpp
28:  *
29:  * contains methods for moving lift based on motor encoders as well as
30:  * with the potentiomter
31:  */
32: class Lift :
33:     public Sensors
34: {
35:     private:
36:
37:     public:
38:         Lift();
39:         ~Lift();
40:
41:         /**
42:          * @param: int revolutions -> the setpoint for the motor to move to
43:          * @return: None
44:          *
45:          * @see: Motors.hpp
46:          *
47:          * moves tilter to within a range of the setpoint
48:          * sets brake mode to brake at the end to act as a holder
49:          */
50:         void move( int revolutions, int timeout=INT32_MAX );
51:
52:         /**
53:          * @param: int pot_value -> the potentiometer setpoint for the motor to move to
54:          * @return: None
55:          *
56:          * @see: Motors.hpp
57:          * @see: Sensors.hpp
58:          *
59:          * moves tilter so that it is at a specific setpoint based on the
60:          * potentiometer
61:          * Useful because the potentiometer acts as an independent reading that will
62:          * be more accurate and consistent
63:          */
64:         void move_to( int pot_value );
65:
66:
67:
68:
69:         /**
70:          * @param: int direction -> move up or down in the setpoints list ( + = up , - = down )
71:          * @return: None
72:          *
73:          * @see: Tilter::move_to
74:          * @see: Configuration.hpp
75:          *
76:          * finds the next or previous index in the list based on the current setpoint
77:          * this index is capped at either the max or min and does not loop around
78:          * no movement is performed if it is at the max or min
79:          * once the new setpoint is determined, move_to is called for the potentiomter value
80:          */
81:         void cycle_setpoint( int direction );
82:
83:
84:         /**
85:          * @param: int target_pot_value -> the setpoint as a potentiometer value to hold at
86:          * @return: None
87:          *
88:          * TODO: add actual pid holding instead of just the brake mode
89:          *
90:          * sets brake mode to pid hold
91:          */
92:         void hold( int target_pot_value ); //holds motors in place
93: };
94:
95:
96:
97: #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/lift/Lift.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/5/19
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: Lift.hpp
8:   *
9:   * contains implementation for the lift subsystem
10:  */
11:
12:  #include <array>
13:  #include <vector>
14:  #include <algorithm>
15:
16:  #include "Lift.hpp"
17:
18:
19:
20:  Lift::Lift()
21:  {
22:
23:  }
24:
25:
26:
27:  Lift::~Lift()
28:  {
29:
30:  }
31:
32:
33:
34:
35:  /**
36:   * moves lift motor to a given setpoint and stops when it is within 10
37:   * encoder ticks
38:   */
39:  void Lift::move( int revolutions, int timeout )
40:  {
41:      Motors *motors = Motors::get_instance();
42:
43:      motors->lift->tare_position();
44:
45:      motors->lift->move_absolute(revolutions, 100);
46:
47:      while ( ( std::abs(revolutions - motors->lift->get_position()) ) > 10 )
48:      {
49:          pros::delay(10);
50:      }
51:
52:      motors->lift->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
53:  }
54:
55:
56:
57:
58:  /**
59:   * moves to a specified potentiomter value and sets brake
60:   * mode to brake to act as a holder
61:   */
62:  void Lift::move_to( int pot_value )
63:  {
64:      Motors *motors = Motors::get_instance();
65:
66:      //TODO: add custom PID controller
67:
68:      motors->lift->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
69:
70:      float error = pot_value - getCorrectedPot();
71:      while ( std::abs( error ) > 10 )
72:      {
73:          // +/- 5 ticks
74:          int revolutions = ( std::abs(error) / error) * 5;  //try catch not needed
75:                                                             //for division because
76:                                                             //segment will not be executed
77:                                                             //if error is 0
78:          motors->lift->move_absolute(revolutions, 100);
79:
80:          while ( motors->lift->get_current_draw() > 0)
81:          {
82:              pros::delay(1);
83:          }
84:
85:          error = pot_value - getCorrectedPot();
86:
87:      }
88:
89:
90:  }
91:
92:
93:
94:
95:  /**
96:   * finds where the current index is in the list of setpoints
97:   * then increases or decreases the index to find the new setpoint
98:   * based on the parameter
99:   * then calls move_to to move to the new setpoint
100:  * caps the max and min and does not wrap back around
101:  */
102: void Lift::cycle_setpoint( int direction )
103: {
104:     Configuration* config = Configuration::get_instance();
105:
106:
107:     Motors *motors = Motors::get_instance();
108:
109:     int current_pot_value = getCorrectedPot();
110:     int target_set_point;
111:
112:     std::vector<int> sorted_setpoints;
113:     for ( int i = 0; i < config->lift_setpoints.size(); i++ )
```

```
114:     {
115:         sorted_setpoints.push_back( config->lift_setpoints[i] );
116:     }
117:     sorted_setpoints.push_back( current_pot_value );
118:
119:     std::sort( sorted_setpoints.begin(), sorted_setpoints.end() );
120:     std::vector<int>::iterator elem = std::find (sorted_setpoints.begin(),
121:                                       sorted_setpoints.end(),
122:                                       current_pot_value);
123:     int index = std::distance(sorted_setpoints.begin(), elem);
124:
125:     int loc = (std::abs(direction) / direction) + index;
126:
127:     if ( !(loc < 0) || !(loc > config->lift_setpoints.size() ))  //cap the max and min for the list
128:     {
129:         target_set_point = config->lift_setpoints[loc];
130:         move_to( target_set_point );
131:     }
132: }
133:
134:
135:
136:
137: /**
138:  * sets brake mode to type brake for the lift
139:  */
140: void Lift::hold( int target_pot_value )
141: {
142:     Motors *motors = Motors::get_instance();
143:
144:     //TODO: add custom PID controller
145:     motors->lift->set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
146:
147: }
148:
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/tilter/Tilter.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/4/19
5:   * @reviewed_by: Aiden Carney
6:   * TODO: Test implementation with robot especially with setpoint and move_to
7:   *
8:   * contains class for tilter sub system and commands to move it
9:   */
10:
11: #ifndef __TILTER_HPP__
12: #define __TILTER_HPP__
13:
14: #include <array>
15:
16: #include "../../../include/main.h"
17: #include "../../../include/pros/motors.hpp"
18:
19: #include "../motors/Motors.hpp"
20: #include "../sensors/Sensors.hpp"
21:
22:
23:
24:
25: /**
26:  * @see: Sensors.hpp
27:  * @see: Motors.hpp
28:  *
29:  * contains methods for moving tilter based on motor encoders as well as
30:  * with the potentiomter
31:  */
32: class Tilter :
33:     public Sensors
34: {
35:     private:
36:
37:     public:
38:         Tilter();
39:         ~Tilter();
40:
41:         /**
42:          * @param: int revolutions -> the setpoint for the motor to move to
43:          * @return: None
44:          *
45:          * @see: Motors.hpp
46:          *
47:          * moves tilter to within a range of the setpoint
48:          * sets brake mode to brake at the end to act as a holder
49:          */
50:         void move( int revolutions );
51:
52:         /**
53:          * @param: int pot_value -> the potentiometer setpoint for the motor to move to
54:          * @return: None
55:          *
56:          * @see: Motors.hpp
57:          * @see: Sensors.hpp
58:          *
59:          * moves tilter so that it is at a specific setpoint based on the
60:          * potentiometer
61:          * Useful because the potentiometer acts as an independent reading that will
62:          * be more accurate and consistent
63:          */
64:         void move_to( int pot_value );
65:
66:
67:
68:
69:         /**
70:          * @param: int direction -> move up or down in the setpoints list ( + = up , - = down )
71:          * @return: None
72:          *
73:          * @see: Tilter::move_to
74:          * @see: Configuration.hpp
75:          *
76:          * finds the next or previous index in the list based on the current setpoint
77:          * this index is capped at either the max or min and does not loop around
78:          * no movement is performed if it is at the max or min
79:          * once the new setpoint is determined, move_to is called for the potentiomter value
80:          */
81:         void cycle_setpoint( int direction );
82:
83:
84:         /**
85:          * @param: int target_pot_value -> the setpoint as a potentiometer value to hold at
86:          * @return: None
87:          *
88:          * TODO: add actual pid holding instead of just the brake mode
89:          *
90:          * sets brake mode to pid hold
91:          */
92:         void hold( int target_pot_value ); //holds motors in place
93: };
94:
95:
96:
97: #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/tilter/Tilter.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/4/19
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: Tilter.hpp
8:   *
9:   * contains implementation for the tilter subsystem
10:  */
11:
12: #include <array>
13: #include <vector>
14: #include <algorithm>
15:
16: #include "Tilter.hpp"
17:
18:
19:
20: Tilter::Tilter()
21: {
22:
23: }
24:
25:
26:
27: Tilter::~Tilter()
28: {
29:
30: }
31:
32:
33:
34:
35: /**
36:  * moves tilter motor to a given setpoint and stops when it is within 10
37:  * encoder ticks
38:  */
39: void Tilter::move( int revolutions )
40: {
41:     Motors *motors = Motors::get_instance();
42:
43:     motors->tilter->tare_position();
44:
45:     motors->tilter->move_absolute(revolutions, 100);
46:
47:     while ( ( std::abs(revolutions - motors->tilter->get_position()) ) > 10 )
48:     {
49:         pros::delay(10);
50:     }
51:
52:     motors->tilter->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
53: }
54:
55:
56:
57:
58: /**
59:  * moves to a specified potentiomter value and sets brake
60:  * mode to brake to act as a holder
61:  */
62: void Tilter::move_to( int pot_value )
63: {
64:     Motors *motors = Motors::get_instance();
65:
66:     //TODO: add custom PID controller
67:
68:     motors->tilter->set_brake_mode(pros::E_MOTOR_BRAKE_BRAKE);
69:
70:     float error = pot_value - getCorrectedPot();
71:     while ( std::abs( error ) > 10 )
72:     {
73:         // +/- 5 ticks
74:         int revolutions = ( std::abs(error) / error) * 5;  //try catch not needed
75:                                                            //for division because
76:                                                            //segment will not be executed
77:                                                            //if error is 0
78:         motors->tilter->move_absolute(revolutions, 100);
79:
80:         while ( motors->tilter->get_current_draw() > 0)
81:         {
82:             pros::delay(1);
83:         }
84:
85:         error = pot_value - getCorrectedPot();
86:
87:     }
88:
89:
90: }
91:
92:
93:
94:
95: /**
96:  * finds where the current index is in the list of setpoints
97:  * then increases or decreases the index to find the new setpoint
98:  * based on the parameter
99:  * then calls move_to to move to the new setpoint
100: * caps the max and min and does not wrap back around
101: */
102: void Tilter::cycle_setpoint( int direction )
103: {
104:     Configuration* config = Configuration::get_instance();
105:
106:
107:     Motors *motors = Motors::get_instance();
108:
109:     int current_pot_value = getCorrectedPot();
110:     int target_set_point;
111:
112:     std::vector<int> sorted_setpoints;
113:     for ( int i = 0; i < config->tilter_setpoints.size(); i++ )
```

```
114:     {
115:         sorted_setpoints.push_back( config->tilter_setpoints[i] );
116:     }
117:     sorted_setpoints.push_back( current_pot_value );
118:
119:     std::sort( sorted_setpoints.begin(), sorted_setpoints.end() );
120:     std::vector<int>::iterator elem = std::find (sorted_setpoints.begin(),
121:                                       sorted_setpoints.end(),
122:                                       current_pot_value);
123:     int index = std::distance(sorted_setpoints.begin(), elem);
124:
125:     int loc = (std::abs(direction) / direction) + index;
126:
127:     if ( !(loc < 0) || !(loc > config->tilter_setpoints.size() ))  //cap the max and min for the list
128:     {
129:         target_set_point = config->tilter_setpoints[loc];
130:         move_to( target_set_point );
131:     }
132: }
133:
134:
135:
136:
137: /**
138:  * sets brake mode to type brake for the tilter
139:  */
140: void Tilter::hold( int target_pot_value )
141: {
142:     Motors *motors = Motors::get_instance();
143:
144:     //TODO: add custom PID controller
145:     motors->tilter->set_brake_mode(pros::E_MOTOR_BRAKE_HOLD);
146:
147: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/writer/Writer.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/4/19
5:   * @reviewed_by: Aiden Carney
6:   * TODO: instead of writing to sd card, flush to serial because sd card is not good
7:   *
8:   * contains class for a writer queue that accepts writes and flushes them
9:   * to an output stream
10:  */
11:
12: #ifndef __WRITER_HPP__
13: #define __WRITER_HPP__
14:
15: #include <atomic>
16: #include <list>
17: #include <queue>
18: #include <string>
19: #include <unordered_map>
20:
21:
22: //#define MACRO_RECORD_DEBUG
23: #define BUFFER_SIZE 256
24:
25:
26: typedef struct
27: {
28:     std::string file;
29:     std::string mode;
30:     std::string content;
31: } writer_obj;
32:
33:
34:
35: /**
36:  * Contains a queue that can be added to and dumped out so that data can
37:  * be gathered and exported
38:  * contains other implementation for interacting with the SD card
39:  */
40: class Writer
41: {
42:     private:
43:         static std::queue<writer_obj> write_queue;
44:         static std::atomic<bool> lock;
45:
46:         static int num_files_open;
47:         static std::string match_file_name;
48:
49:         /**
50:          * @return: writer_obj
51:          *
52:          * gets an object from the writer queue
53:          * returns an uninitialized writer_obj if the queue is empty
54:          */
55:         writer_obj get_write( );
56:
57:         /**
58:          * @param: std::string file_name -> location of the file
59:          * @param: std::string mode -> mode to open file in (a for append, w for write)
60:          * @param: std::string contents -> what to write to the file
61:          * @return: bool -> true if the file was actually written to, false if an error occured
62:          *
63:          * uses the c api to take a big string and write BUFFER_SIZE chars to
64:          * file at a time
65:          *
66:          * currently writes to a big match file that would have to be parsed
67:          * segfault occurs when writing with errno 5, so this command should not
68:          * be used in a match
69:          */
70:         bool write( std::string file_name, std::string mode, std::string contents );
71:
72:     public:
73:         Writer();
74:         ~Writer();
75:
76:         /**
77:          * @param: writer_obj test_item -> item to add to the writer queue
78:          * @return: None
79:          *
80:          * adds an item to the writer queue
81:          * the queue is protected using a spinlock implemented with an
82:          * std::atomic bool
83:          */
84:         void add( writer_obj test_item );
85:
86:         /**
87:          * @return: bool -> true on success and false if an error occured in the process
88:          *
89:          * builds up a cache of items from teh queue for 50ms so that they can be
90:          * written closer to the max file write speed
91:          * currently uses private write method which will segfault under high loads
92:          * do not use during a competition
93:          */
94:         bool dump( );
95:
96:
97:
98:
99:         /**
100:         * @param: std::string prev_name -> name of the file to change the name of
101:         * @param: std::string new_name -> new name for the file
102:         * @return: None
103:         *
104:         * coes not actually rename the file as that has not been implented in the
105:         * vex os or pros api
106:         * instead writes contents of current file to the file of the new name
107:         */
108:        void rename( std::string prev_name, std::string new_name);
109:
110:        /**
111:         * @param: std::string file_name
112:         * @return: None
113:         *
```

```
114:         * coes not actually remove the file as that has not been implented in the
115:         * vex os or pros api
116:         * instead opens the file in truncate mode to clear the contens of the file
117:         * allows for mock sd card functionality
118:         */
119:        void remove( std::string file_name );
120:
121:
122:
123:
124:        /**
125:         * @return: int -> number of items in the writer queue
126:         *
127:         * returns the size of the writer queue
128:         */
129:        static int get_count();
130:
131:        /**
132:         * @return: int -> number of files tracked to be open
133:         *
134:         * returns the number of files open
135:         * this is not a full proof method as this is only kept track of
136:         * when a file is opened using the write method in the class
137:         * because of this, this is mainly to be used for debugging
138:         */
139:        static int get_num_files_open();
140:    };
141:
142:
143:
144:    #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/writer/Writer.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 12/5/19
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see Writer.hpp
8:   *
9:   * contains implementation for the writer class
10:  */
11:
12:  #include <atomic>
13:  #include <chrono>
14:  #include <ctime>
15:  #include <ctype.h>
16:  #include <fstream>
17:  #include <iostream>
18:  #include <list>
19:  #include <queue>
20:  #include <sstream>
21:  #include <string>
22:  #include <assert.h>
23:  #include <iostream>
24:  #include <cmath>
25:  #include <cerrno>
26:  #include <cstring>
27:  #include <clocale>
28:
29:
30:  #include "../../../include/main.h"
31:
32:  #include "../../../lib/date/date.h"
33:  #include "Writer.hpp"
34:
35:  std::queue<writer_obj> Writer::write_queue;
36:  std::atomic<bool> Writer::lock = ATOMIC_VAR_INIT(false);
37:  int Writer::num_files_open = 0;
38:  std::string Writer::match_file_name = "/usd/match_data/" + date::format("%d-%m-%Y", date::floor<std::chrono::microseconds>(std::chrono::system_clock::now())) + ".match";
39:
40:
41:  Writer::Writer() { }
42:
43:
44:
45:
46:  Writer::~Writer() { }
47:
48:
49:
50:
51:  /**
52:   * formats content to be written to a match file to avoid having multiplte files open
53:   * writes data by making chunks of 256 bytes
54:   * uses the c api and performs checks to see where errors occur
55:   * typically sets errno to 5: too many files open in system even though only
56:   * one file has ever been opened
57:   */
58:  bool Writer::write( std::string file_name, std::string mode, std::string contents )
59:  {
60:      std::string to_write = "{'filename':'" + file_name + "','mode':'" + mode + "','contents':'";
61:      int i = 0;
62:      FILE *f;
63:      int size;
64:      int written;
65:
66:      f = fopen(match_file_name.c_str(), "a");
67:      if ( f == NULL )  //check file was opened correctly
68:      {
69:          std::cout << "not opened correctly\n" << std::flush;
70:          std::cout << "errno: " << errno << "\n" << std::flush;
71:          std::cout << "strerror: " << std::strerror(errno) << "\n" << std::flush;
72:          std::cout << match_file_name.c_str() << "\n" << std::flush;
73:          pros::delay(10);
74:          fclose(f);
75:          return false;  //nothing was written
76:      }
77:      num_files_open += 1;
78:
79:
80:      //finds all '\n' characters and replaces with /*eol*/ so it can be parsed more consistently
81:      std::string to = "/*eol*/";
82:      std::string from = "\n";
83:      size_t start_pos = 0;
84:      while((start_pos = contents.find(from, start_pos)) != std::string::npos) {
85:          contents.replace(start_pos, from.length(), to);
86:          start_pos += to.length(); // Handles case where 'to' is a substring of 'from'
87:      }
88:      to_write += contents + "'}\n";
89:
90:
91:      size = to_write.size();
92:      written = 0;
93:      while ( written < size )
94:      {
95:          char tempBuf[BUFFER_SIZE];
96:
97:          for( i=0; i<(BUFFER_SIZE - 1); i++ )  //use BUFFER_SIZE - 1 so that terminator can be added
98:          {
99:              if ( written >= (size) )  //use ">=" so that no characters past the size of the string will be printed
100:             {
101:                 break;
102:             }
103:             tempBuf[i] = to_write.at(written);
104:             written += 1;
105:         }
106:         tempBuf[i] = '\0';
107:
108:         std::cout << "writing to file\n";
109:
110:         int ret = fputs(tempBuf, f);  //write to file
111:         std::cout << "errno: " << errno << "\n" << std::flush;
112:         std::cout << "strerror: " << std::strerror(errno) << "\n" << std::flush;
113:         pros::delay(6);
```

```
114:        fflush(f);
115:
116:        //int ret = fwrite ( tempBuf, sizeof(char), sizeof(tempBuf), f);
117:        if ( ret == EOF )
118:        {
119:            std::cout << "not written correctly\n" << std::flush;
120:            std::cout << "errno: " << errno << "\n" << std::flush;
121:            std::cout << "strerror: " << std::strerror(errno) << "\n" << std::flush;
122:            std::cout << file_name.c_str() << "\n" << std::flush;
123:            pros::delay(10);
124:            fclose(f);
125:            return false;  //operation failed
126:        }
127:        //std::cout << "items written: " << written << "  items to write: " << size << "  items left: " << (size - written) << "\n";
128:    }
129:
130:    std::cout << "closing file\n";
131:    pros::delay(5);
132:
133:    fflush(f);
134:    int ret = fclose(f);
135:    if ( ret == EOF )
136:    {
137:        std::cout << "not closed correctly\n" << std::flush;
138:        std::cout << "errno: " << errno << "\n" << std::flush;
139:        std::cout << "strerror: " << std::strerror(errno) << "\n" << std::flush;
140:        std::cout << file_name.c_str() << "\n" << std::flush;
141:        pros::delay(10);
142:    }
143:    else
144:    {
145:        num_files_open -= 1;
146:    }
147:    std::cout << "finished\n";
148:    std::cout << "errno: " << errno << "\n" << std::flush;
149:    std::cout << "strerror: " << std::strerror(errno) << "\n" << std::flush;
150:    return true;
151:
152: }
153:
154:
155:
156:
157: /**
158:  * add item to the writer queue by aquiring and releasing atomic lock
159:  */
160: void Writer::add( writer_obj test_item )
161: {
162:    if ( !test_item.file.empty() && !test_item.mode.empty() )
163:    {
164:        while ( lock.exchange( true ) ); //aquire lock
165:        write_queue.push( test_item );
166:        lock.exchange( false ); //release lock
167:    }
168:    else
169:    {
170:        std::cout << "adding failed********************\n";
171:    }
172: }
173:
174:
175:
176:
177: /**
178:  * gets an item from the queue by acquiring the lock and releasing it
179:  */
180: writer_obj Writer::get_write( )
181: {
182:    writer_obj contents;
183:
184:    //checks if there are items to be written, useful so that this function is
185:    //not blocking if the queue is empty
186:    if ( !write_queue.empty() )
187:    {
188:        while ( lock.exchange( true ) ); //aquire lock
189:        contents = write_queue.front();
190:        write_queue.pop();
191:        lock.exchange( false ); //release lock because there is no more iteraction
192:                                //with the queue
193:
194:    }
195:    return contents;
196: }
197:
198:
199:
200:
201: /**
202:  * builds up a cache of items based on the file name and writes it in bulk
203:  * the items are stored in vectors based on the file name
204:  * this is used so that the max write speed can be used and less time is spent
205:  * waiting
206:  */
207: bool Writer::dump( )
208: {
209:    std::vector< writer_obj > writes;
210:    std::vector< writer_obj > temp_vec;
211:
212:    writes.reserve(100000);
213:    temp_vec.reserve(100000);
214:
215:    int stop = pros::millis() + 50;
216:    //for ( int i = 0; i < 200; i++ )  //collects up to 200 writer objects
217:    while ( pros::millis() < stop )
218:    {
219:        writer_obj contents = get_write();
220:
221:        if ( !contents.file.empty() && !contents.mode.empty() )
222:        {
223: /*         std::ofstream out_file( contents.file, std::ios_base::app ); //dump current contents
224:            num_files_open += 1;
225:            out_file << contents.content;
226:            out_file.close();
```

```
227:            num_files_open -= 1;*/
228:            writes.push_back( contents );
229:        }
230:        else //object returned was empty meaning queue is empty
231:            //it would be wasteful to look at the queue when it is known to be empty
232:        {
233:            pros::delay(1);
234:            //break;
235:        }
236:    }
237:
238:    if ( writes.empty() )
239:    {
240:        return false;
241:    }
242:
243:    while ( !writes.empty() )
244:    {
245:        std::string write_loc;
246:        std::string write_mode;
247:        std::string write_contents;
248:        #ifdef MACRO_RECORD_DEBUG
249:            std::string write_contents_debug;
250:            std::string delimiter = ",";
251:        #endif
252:
253:        write_contents.reserve(2500);
254:        #ifdef MACRO_RECORD_DEBUG
255:            write_contents_debug.reserve(10000);
256:        #endif
257:
258:
259:        int first = 0;
260:        while ( write_mode.empty() && ( first < writes.size() ) )  //iterate until the first element is not of mode write
261:                                                                   //algorithm writes mode "w" immediately
262:        {
263:            if ( writes.at(first).mode.compare("a") == 0 )
264:            {
265:                write_loc = writes.at(first).file;
266:                write_mode = writes.at(first).mode;
267:                write_contents = writes.at(first).content;
268:                #ifdef MACRO_RECORD_DEBUG
269:                    write_contents_debug = (writes.at(first).content.substr(0, writes.at(first).content.find(delimiter))
270:                        + ","
271:                        + date::format("%d-%m-%Y %T", date::floor<std::chrono::microseconds>(std::chrono::system_clock::now()))
272:                        + "\n");
273:                #endif
274:                break;
275:            }
276:            else
277:            {
278:                write(writes.at(first).file, "w", writes.at(first).content);
279:            }
280:            first += 1;
281:        }
282:
283:
284:        for ( int i = first; i < writes.size(); i++ )
285:        {
286:            //case 1: the object in question is mode append and is the file that is being
287:            //searched for
288:            if ( writes.at(i).file.compare(write_loc) == 0 && writes.at(i).mode.compare("a") == 0 )
289:            {
290:                write_contents += writes.at(i).content;
291:                #ifdef MACRO_RECORD_DEBUG
292:                    write_contents_debug += (writes.at(i).content.substr(0, writes.at(i).content.find(delimiter))
293:                        + ","
294:                        + date::format("%d-%m-%Y %T", date::floor<std::chrono::microseconds>(std::chrono::system_clock::now()))
295:                        + "\n");
296:                #endif
297:            }
298:            //case 2: the object in question is the file that is being searched for but is of
299:            //mode write
300:            else if ( writes.at(i).file.compare(write_loc) == 0 && writes.at(i).mode.compare("w") == 0 )
301:            {
302:                write(write_loc, "a", write_contents);
303:
304:                #ifdef MACRO_RECORD_DEBUG
305:                    write("/usd/log/15/from_queue/data.csv", "a", write_contents_debug);
306:                #endif
307:
308:                write(write_loc, "w", writes.at(i).content);
309:
310:                write_loc.clear();
311:                write_mode.clear();
312:                write_contents.clear();
313:
314:                break;
315:            }
316:            //case 3 the object in question is not of the same file
317:            else
318:            {
319:                temp_vec.push_back( writes.at(i) );
320:            }
321:
322:        }
323:    /*      //std::cout << write_contents << "\n\n";
324:        std::ofstream out_file( write_loc, std::ios_base::app ); //dump current contents
325:        num_files_open += 1;
326:        if ( out_file.good() )
327:        {
328:            out_file << write_contents;
329:        }
330:        else
331:        {
332:            std::cout << "file not opened correctly\n";
333:        }
334:        out_file.close();
335:        num_files_open -= 1;*/
336:        //std::cout << "starting write\n";
337:        bool ret = write(write_loc, "a", write_contents);
338:        if (!ret)
339:        {
```

```
340:             std::cout << "write failed\n";
341:             pros::delay(5);
342:         }
343:         //std::cout << "write finished\n";
344:
345:         #ifdef MACRO_RECORD_DEBUG
346:             write("/usd/log/15/from_queue/data.csv", "a", write_contents_debug);
347:         #endif
348:
349:         writes.clear();  //swap contents of both lists
350:
351:         std::cout << writes.size() << " " << temp_vec.size() << "\n";
352:
353:         for ( int i = 0; i < temp_vec.size(); i++ )
354:         {
355:             writes.push_back( temp_vec.at(i) );
356:         }
357:         temp_vec.clear();
358:
359:     }
360:
361:     return true;
362:
363: }
364:
365:
366:
367:
368: /**
369:  * opens file and copies it to the new file name so that is appears to have
370:  * been renamed
371:  * also functions as a file move operation
372:  */
373: void Writer::rename( std::string prev_name, std::string new_name)
374: {
375:     //perform operation with lock aquired because it is a copy
376:     //rename
377:     while ( lock.exchange( true ) ); //aquire lock
378:
379:     std::ifstream src;
380:     std::ofstream dst;
381:     num_files_open += 1;
382:
383:     src.open(prev_name, std::ios::in);
384:     dst.open(new_name, std::ios::app);
385:
386:     dst << src.rdbuf();
387:
388:     src.close();
389:     num_files_open -= 1;
390:     dst.close();
391:
392:     lock.exchange( false ); //release lock
393: }
394:
395:
396:
397:
398: /**
399:  * simulates removing a file by opening it in truncate mode
400:  * which clears the contents of the file
401:  */
402: void Writer::remove( std::string file_name )
403: {
404:     //perform operation with lock aquired because it is a clear contents
405:     //delete
406:     while ( lock.exchange( true ) ); //aquire lock
407:
408:     std::ofstream file( file_name, std::ios::out | std::ios::trunc ); // clear contents
409:     num_files_open += 1;
410:     file.close( );
411:     num_files_open -= 1;
412:
413:     lock.exchange( false ); //release lock
414: }
415:
416:
417:
418:
419: /**
420:  * gets the size of the writer queue
421:  */
422: int Writer::get_count()
423: {
424:     return write_queue.size();
425: }
426:
427:
428:
429:
430: /**
431:  * gets the number of files opened that has been tracked by methods in the class
432:  */
433: int Writer::get_num_files_open()
434: {
435:     return num_files_open;
436: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/lcdCode/Gimmicks.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: fix loading screen, it sometimes does not work
7:   *
8:   * contains lcd gimmicks that are used to enhance interface
9:   *
10:  */
11:
12:  #ifndef __GIMMICKS_HPP__
13:  #define __GIMMICKS_HPP__
14:
15:  #include <string>
16:
17:  #include "../../../include/main.h"
18:
19:  #include "Styles.hpp"
20:
21:
22:  /**
23:   * @see: Styles.hpp
24:   * @see: ./lcdCode
25:   *
26:   * used to display warning box
27:   */
28:  class WarningMessage : virtual Styles
29:  {
30:      protected:
31:          /**
32:           * @param: lv_obj_t* mbox -> message box object
33:           * @param: const char* txt -> text for message box
34:           * @return: LV_RES_OK -> if finishes successfully
35:           *
36:           * @see: Styles.hpp
37:           * @see: ./lcdCode
38:           *
39:           * sets static int option to positive or negative based on feedback
40:           *
41:           */
42:          static lv_res_t mbox_apply_action(lv_obj_t * mbox, const char * txt);
43:          static const char* buttons[];
44:          static int option;
45:
46:          lv_obj_t *warn_box;
47:
48:
49:      public:
50:          WarningMessage();
51:          virtual ~WarningMessage();
52:
53:          /**
54:           * @param: std::string warn_msg -> message that will appear as option
55:           * @param: lv_obj_t* parent -> the parent that the message box will appear on
56:           * @return: bool -> if user selected yes or no
57:           *
58:           * returns true or false based on what user selects
59:           * implementation of this is up to user
60:           *
61:           */
62:          bool warn(std::string warn_msg, lv_obj_t *parent);
63:
64:  };
65:
66:
67:  /**
68:   * @see: Styles.hpp
69:   * @see: ./lcdCode
70:   *
71:   * methods and objects for a loading bar
72:   */
73:  class Loading : virtual Styles
74:  {
75:      protected:
76:          lv_obj_t *loader;
77:
78:      public:
79:          Loading();
80:          ~Loading();
81:
82:          /**
83:           * @param: int estimated_duration -> duration that loading should take used to set speed of bar
84:           * @param: lv_obj_t* parent -> parent object that loading bar will go on
85:           * @param: int x -> x position of loading bar relative to parent
86:           * @param int y -> y position of loading bar relative to parent
87:           * @return: None
88:           *
89:           * shows the loader and starts the action of it moving
90:           *
91:           */
92:          void show_load(int estimated_duration, lv_obj_t *parent, int x, int y); //starts the loader
93:
94:          /**
95:           * @return: None
96:           *
97:           * hides the loader
98:           * this should be about when the loader is finished
99:           * Used to keep a smooth transition
100:          *
101:          */
102:         void hide_load(); //ends the loader and hides it
103: };
104:
105:
106:
107:
108:
109:
110:
111: #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/lcdCode/Gimmicks.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: Gimmicks.hpp
8:   *
9:   * contains implementation for header file
10:  *
11:  */
12:
13: #include "../../../include/main.h"
14: #include "../../../include/api.h"
15:
16: #include "Styles.hpp"
17: #include "Gimmicks.hpp"
18:
19:
20:
21: const char* WarningMessage::buttons[] = {"Back", "Continue", ""};
22:
23: //base classes
24: int WarningMessage::option = 0;
25:
26: WarningMessage::WarningMessage()
27: {
28:     option = 0;
29:
30:     warn_box = lv_mbox_create(lv_scr_act(), NULL);
31:     lv_mbox_set_text(warn_box, "None");
32:     lv_mbox_add_btns(warn_box, buttons, NULL);
33:     lv_mbox_set_action(warn_box, mbox_apply_action);
34:
35:     lv_mbox_set_style(warn_box, LV_MBOX_STYLE_BG, &warn_box_bg);
36:     lv_mbox_set_style(warn_box, LV_MBOX_STYLE_BTN_REL, &warn_box_released);
37:     lv_mbox_set_style(warn_box, LV_MBOX_STYLE_BTN_PR, &warn_box_pressed);
38:
39:     lv_obj_set_width(warn_box, 400);
40:     lv_obj_set_height(warn_box, 140);
41:
42:     lv_obj_align(warn_box, NULL, LV_ALIGN_CENTER, 0, -50);
43:
44:
45: }
46:
47: WarningMessage::~WarningMessage()
48: {
49:     lv_obj_del(warn_box);
50: }
51:
52: /**
53:  * compares text of message to set option to positive or negative
54:  */
55: lv_res_t WarningMessage::mbox_apply_action(lv_obj_t * mbox, const char * txt)
56: {
57:     if ( txt == "Continue" )
58:     {
59:         option = 1;
60:     }
61:
62:     else if ( txt == "Back" )
63:     {
64:         option = -1;
65:     }
66:
67:     return LV_RES_OK;
68: }
69:
70: /**
71:  * dislays a message box and sets the text
72:  * user can choose "continue" or "back"
73:  * how function works line 3
74:  */
75: bool WarningMessage::warn( std::string warn_msg, lv_obj_t *parent )
76: {
77:     option = 0;
78:
79:     lv_obj_set_hidden(warn_box, false);
80:     lv_obj_set_parent(warn_box, parent);
81:     lv_mbox_set_text(warn_box, warn_msg.c_str());
82:
83:     while ( !(option) )
84:     {
85:         pros::delay(50);
86:     }
87:
88:     if ( option == 1 )
89:     {
90:         lv_obj_set_hidden(warn_box, true);
91:         return true;
92:     }
93:
94:     else
95:     {
96:         lv_obj_set_hidden(warn_box, true);
97:         return false;
98:     }
99:
100: }
101:
102:
103:
104:
105:
106:
107: Loading::Loading()
108: {
109:     loader = lv_bar_create(lv_scr_act(), NULL);
110:     lv_obj_set_size(loader, 100, 20);
111:     lv_bar_set_value(loader, 1);
112: }
113:
```

```
114: Loading::~Loading()
115: {
116:     lv_obj_del(loader);
117: }
118:
119: /**
120:  * loader is shown on a parent at specified location
121:  * animation time is set by user, so this function only works if user knows
122:  * about how long the function will take
123:  * this function is meant as a filler so that if some initialization occurs
124:  * the gui does not appear like its hanging for no reason
125:  */
126: void Loading::show_load(int estimated_duration, lv_obj_t *parent, int x, int y)
127: {
128:     lv_obj_set_hidden(loader, false);
129:     lv_bar_set_value(loader, 1);
130:     lv_obj_set_parent(loader, parent);
131:     lv_obj_set_top(loader, true);
132:
133:     lv_obj_set_pos(loader, x, y);
134:
135:     lv_bar_set_value_anim(loader, 100, estimated_duration);
136: }
137:
138: /**
139:  * hides the loader for when the initialization by user is finished
140:  */
141: void Loading::hide_load()
142: {
143:     lv_obj_set_hidden(loader, true);
144: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/lcdCode/Styles.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * contains base class for styles of gui objects
8:   *
9:   */
10:
11: #ifndef __STYLES__
12: #define __STYLES__
13:
14:
15: #include "../../../include/main.h"
16:
17: //defines colors to use for each style
18: #define BLUE_BORDER LV_COLOR_BLUE
19: #define RED_BORDER LV_COLOR_RED
20: #define BG LV_COLOR_GRAY
21: #define BUTTON_REL LV_COLOR_SILVER
22: #define BUTTON_PR LV_COLOR_NAVY
23: #define TEXT LV_COLOR_WHITE
24: #define BODY_TEXT LV_COLOR_BLACK
25: #define SW_INDIC LV_COLOR_HEX(0x9fc8ef)
26:
27: //allows use of other fonts
28: #define USE_DEJAVU_12
29: #define USE_DEJAVU_16
30: #include "../../../include/fonts/fonts.h"
31:
32:
33: /**
34:  * @see: ../../include/fonts/fonts.hpp
35:  * @see ../fonts/
36:  *
37:  * base class that contains different colors and styles to be used throughout
38:  * the gui
39:  * designed so that there is no repetion of styles and so they are all in one place
40:  * designed to be inherited
41:  */
42: class Styles
43: {
44:     protected:
45:         //styles
46:         lv_style_t blue;
47:         lv_style_t red;
48:         lv_style_t gray;
49:
50:         lv_style_t toggle_btn_released;
51:         lv_style_t toggle_btn_pressed;
52:
53:         lv_style_t toggle_tabbtn_released;
54:         lv_style_t toggle_tabbtn_pressed;
55:
56:         lv_style_t sw_toggled;
57:         lv_style_t sw_off;
58:         lv_style_t sw_bg;
59:         lv_style_t sw_indic;
60:
61:         lv_style_t heading_text;
62:         lv_style_t body_text;
63:         lv_style_t subheading_text;
64:
65:         lv_style_t lines;
66:
67:         lv_style_t warn_box_bg;
68:         lv_style_t warn_box_pressed;
69:         lv_style_t warn_box_released;
70:
71:         lv_style_t loader_style;
72:
73:     public:
74:         Styles();
75:         virtual ~Styles();
76:
77: };
78:
79: #endif
```

```
 1: /**
 2:  * @file: ./RobotCode/src/objects/lcdCode/Styles.cpp
 3:  * @author: Aiden Carney
 4:  * @reviewed_on: 10/15/2019
 5:  * @reviewed_by: Aiden Carney
 6:  *
 7:  * @see: Styles.hpp
 8:  *
 9:  * contains base class for styles for gui
10:  *
11:  */
12:
13: #include "../../../include/main.h"
14: #include "../../../include/api.h"
15:
16: #include "Styles.hpp"
17:
18:
19: Styles::Styles()
20: {
21: //red style
22:     lv_style_copy(&red, &lv_style_scr);
23:     red.body.main_color = LV_COLOR_RED;
24:     red.body.grad_color = LV_COLOR_RED;
25:     red.body.border.color = LV_COLOR_RED;
26:
27: //blue style
28:     lv_style_copy(&blue, &lv_style_scr);
29:     blue.body.main_color = LV_COLOR_BLUE;
30:     blue.body.grad_color = LV_COLOR_BLUE;
31:     blue.body.border.color = LV_COLOR_BLUE;
32:
33: //gray style
34:     lv_style_copy(&gray, &lv_style_scr);
35:     gray.body.main_color = BG;
36:     gray.body.grad_color = BG;
37:     gray.body.border.color = BG;
38:     gray.body.border.width = 10;
39:
40: //style for when the button is not pressed
41:     lv_style_copy(&toggle_btn_released, &lv_style_plain);
42:     toggle_btn_released.body.main_color = BUTTON_REL;
43:     toggle_btn_released.body.grad_color = BUTTON_REL;
44:     toggle_btn_released.body.border.color = BUTTON_REL;
45:     toggle_btn_released.body.border.width = 2;
46:     toggle_btn_released.body.border.opa = LV_OPA_0;
47:     toggle_btn_released.body.radius = 5;
48:     toggle_btn_released.text.color = TEXT;
49:
50: //style for when the button is pressed
51:     lv_style_copy(&toggle_btn_pressed, &lv_style_plain);
52:     toggle_btn_pressed.body.main_color = BUTTON_PR;
53:     toggle_btn_pressed.body.grad_color = BUTTON_PR;
54:     toggle_btn_pressed.body.border.color = BUTTON_REL;
55:     toggle_btn_pressed.text.color = TEXT;
56:
57: //style for when tabview button is not pressed
58:     lv_style_copy(&toggle_tabbtn_released, &lv_style_plain);
59:     toggle_tabbtn_released.body.main_color = BUTTON_REL;
60:     toggle_tabbtn_released.body.grad_color = BUTTON_REL;
61:     toggle_tabbtn_released.body.border.color = BUTTON_REL;
62:     toggle_tabbtn_released.body.border.width = 2;
63:     toggle_tabbtn_released.body.border.opa = LV_OPA_0;
64:     toggle_tabbtn_released.text.color = TEXT;
65:     toggle_tabbtn_released.text.font = &dejavu_12;
66:
67: //style for when tabview button is pressed
68:     lv_style_copy(&toggle_tabbtn_pressed, &lv_style_plain);
69:     toggle_tabbtn_pressed.body.main_color = BUTTON_PR;
70:     toggle_tabbtn_pressed.body.grad_color = BUTTON_PR;
71:     toggle_tabbtn_pressed.body.border.color = BUTTON_REL;
72:     toggle_tabbtn_pressed.text.color = TEXT;
73:     toggle_tabbtn_pressed.text.font = &dejavu_12;
74:
75: //switch on
76:     lv_style_copy(&sw_toggled, &lv_style_pretty_color);
77:     sw_toggled.body.radius = LV_RADIUS_CIRCLE;
78:     sw_toggled.body.shadow.width = 4;
79:     sw_toggled.body.shadow.type = LV_SHADOW_BOTTOM;
80:
81: //switch off
82:     lv_style_copy(&sw_off, &lv_style_pretty);
83:     sw_off.body.radius = LV_RADIUS_CIRCLE;
84:     sw_off.body.shadow.width = 4;
85:     sw_off.body.shadow.type = LV_SHADOW_BOTTOM;
86:
87: //switch background
88:     lv_style_copy(&sw_bg, &lv_style_pretty);
89:     sw_bg.body.radius = LV_RADIUS_CIRCLE;
90:
91: //switch indicator
92:     lv_style_copy(&sw_indic, &lv_style_pretty_color);
93:     sw_indic.body.radius = LV_RADIUS_CIRCLE;
94:     sw_indic.body.main_color = SW_INDIC;
95:     sw_indic.body.grad_color = SW_INDIC;
96:     sw_indic.body.padding.hor = 0;
97:     sw_indic.body.padding.ver = 0;
98:
99: //heading text
100:     lv_style_copy(&heading_text, &lv_style_plain);
101:     heading_text.text.letter_space = 2;
102:     heading_text.text.line_space = 1;
103:     heading_text.text.color = TEXT;
104:     heading_text.text.font = &lv_font_dejavu_20;
105:
106: //body text
107:     lv_style_copy(&body_text, &lv_style_plain);
108:     body_text.text.letter_space = 2;
109:     body_text.text.line_space = 1;
110:     body_text.text.color = BODY_TEXT;
111:     body_text.text.font = &dejavu_12;
112:
113: //subheading text
```

```
114:     lv_style_copy(&subheading_text, &lv_style_plain);
115:     subheading_text.text.letter_space = 2;
116:     subheading_text.text.line_space = 1;
117:     subheading_text.text.color = BODY_TEXT;
118:     subheading_text.text.font = &dejavu_16;
119:
120:   //style for lines
121:     lv_style_copy(&lines, &lv_style_plain);
122:     lines.line.color = BUTTON_PR;
123:     lines.line.width = 5;
124:
125:   //styles for warning box
126:     //background
127:     lv_style_copy(&warn_box_bg, &lv_style_pretty);
128:     warn_box_bg.body.main_color = LV_COLOR_MAKE(0xf5, 0x45, 0x2e);
129:     warn_box_bg.body.grad_color = LV_COLOR_MAKE(0xb9, 0x1d, 0x09);
130:     warn_box_bg.body.border.color = LV_COLOR_MAKE(0x3f, 0x0a, 0x03);
131:     warn_box_bg.text.color = LV_COLOR_WHITE;
132:     warn_box_bg.body.padding.hor = 12;
133:     warn_box_bg.body.padding.ver = 8;
134:     warn_box_bg.body.shadow.width = 8;
135:
136:     //button not pressed
137:     lv_style_copy(&warn_box_released, &lv_style_btn_rel);
138:     warn_box_released.body.empty = 1;
139:     warn_box_released.body.border.color = LV_COLOR_WHITE;
140:     warn_box_released.body.border.width = 2;
141:     warn_box_released.body.border.opa = LV_OPA_70;
142:     warn_box_released.body.padding.hor = 12;
143:     warn_box_released.body.padding.ver = 8;
144:
145:     //button being pressed
146:     lv_style_copy(&warn_box_pressed, &warn_box_released);
147:     warn_box_pressed.body.empty = 0;
148:     warn_box_pressed.body.main_color = LV_COLOR_MAKE(0x5d, 0x0f, 0x04);
149:     warn_box_pressed.body.grad_color = LV_COLOR_MAKE(0x5d, 0x0f, 0x04);
150:
151:   //style for loader
152:     lv_style_copy(&loader_style, &lv_style_plain);
153:     loader_style.line.width = 10;  //10 px thick arc
154:     loader_style.line.color = LV_COLOR_HEX3(0x258); //Blueish arc color
155:
156:     loader_style.body.border.color = LV_COLOR_HEX3(0xBBB); //Gray background color
157:     loader_style.body.border.width = 10;
158:     loader_style.body.padding.hor = 0;
159:
160:   }
161:
162:   Styles::~Styles()
163:   {
164:
165:   }
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/lcdCode/TemporaryScreen.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: deprecate, possibly move somewhere else, file does very little and could be merged elsewhere
7:   *
8:   * contains a global static screen that can be loaded so that the one screen needs to
9:   * be loaded at all times rule is not broken
10:  *
11:  */
12:
13: #ifndef __TEMPORARYSCREEN_HPP__
14: #define __TEMPORARYSCREEN_HPP__
15:
16: #include "../../../include/main.h"
17:
18:
19: struct tempScreen
20: {
21:     static lv_obj_t *temp_screen;
22: };
23:
24:
25:
26: #endif
```

```
 1: /**
 2:  * @file: ./RobotCode/src/objects/lcdCode/TemporaryScreen.cpp
 3:  * @author: Aiden Carney
 4:  * @reviewed_on: 10/15/2019
 5:  * @reviewed_by: Aiden Carney
 6:  *
 7:  * @see: TemporaryScreen.hpp
 8:  *
 9:  * global screen part of a struct that can be loaded
10:  * has no parent so that it is always valid
11:  *
12:  */
13:
14: #include "TemporaryScreen.hpp"
15: #include "Styles.hpp"
16: #include "../../../include/main.h"
17:
18:
19: lv_obj_t *tempScreen::temp_screen = lv_obj_create(NULL, NULL);
```

```
1:  /**
2:   * @file: ./RobotCode/src/objects/lcdCode/gui.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: clean up conditionals, add config file
7:   *
8:   * contains auton selector gui selection all put together in one function
9:   *
10:  */
11: #ifndef __GUI_HPP__
12: #define __GUI_HPP__
13:
14:
15: #include "../include/main.h"
16:
17: #include "AutonSelection/SelectionScreen.hpp"
18: #include "AutonSelection/OptionsScreen.hpp"
19: #include "AutonSelection/PrepScreen.hpp"
20: #include "AutonSelection/ActionsScreen.hpp"
21: #include "DriverControl/DriverControlLCD.hpp"
22: #include "../../DriverControl.hpp"
23: #include "Debug/Debug.hpp"
24: #include "TemporaryScreen.hpp"
25:
26:
27: /**
28:  * @return: int -> number of auton selected
29:  *
30:  * @see: ./AutonSelection
31:  * @see: ./Debug
32:  *
33:  * TODO: add more meaningful config options, clean up conditionals
34:  *
35:  * iterates and interacts with user to find final auton choice, and config options
36:  *
37:  */
38: int chooseAuton();
39:
40:
41:
42:
43:
44: #endif
```

```
  1: /**
  2:  * @file: ./RobotCode/src/objects/lcdCode/gui.cpp
  3:  * @author: Aiden Carney
  4:  * @reviewed_on: 10/25/2019
  5:  * @reviewed_by: Aiden Carney
  6:  *
  7:  * @see: gui.hpp
  8:  *
  9:  * contains implementation of gui
 10:  *
 11:  */
 12:
 13: #include "../../../include/main.h"
 14:
 15: #include "gui.hpp"
 16: #include "../../Autons.hpp"
 17: #include "../motors/MotorThread.hpp"
 18: #include "../../DriverControl.hpp"
 19: #include "TemporaryScreen.hpp"
 20:
 21:
 22: /**
 23:  * iterates through selecting for user to go through stages selecting an auton or the debugger
 24:  * and then all the config options
 25:  * loads all screens at start so there are no mem management issues
 26:  * finishes when all options are chosen
 27:  */
 28: int chooseAuton()
 29: {
 30:     Autons auton_data;
 31:
 32:     //init screens so that loading time is faster
 33:     SelectionScreen scr1;
 34:     OptionsScreen scr2;
 35:     PrepScreen scr3;
 36:     DriverControlLCD scr4;
 37:
 38:     int finalAutonChoice = 0;
 39:     int auton = 1;
 40:     bool confirm = false;
 41:     int interval = 20;
 42:
 43:     while ( !(finalAutonChoice) ) //allows user to go to previous screen
 44:     {
 45:         scr2.back = false;
 46:
 47:         auton = scr1.selectAuton( auton ); //get auton option
 48:
 49:         if ( auton == auton_data.driver_control_num ) //if prog with no auton is selected
 50:         {
 51:             finalAutonChoice = 1;
 52:         }
 53:
 54:         else if ( auton == auton_data.debug_auton_num ) //if debugger is selected
 55:         {
 56:             //starts driver control for debugging purposes
 57:             pros::Task driver_control_task (driver_control,
 58:                                             (void*)NULL,
 59:                                             TASK_PRIORITY_DEFAULT,
 60:                                             TASK_STACK_DEPTH_DEFAULT,
 61:                                             "DriverControlTask");
 62:
 63:             debug();
 64:
 65:             //ends driver control because it should not be enabled when
 66:             //auton is being selected
 67:             driver_control_task.remove();
 68:         }
 69:
 70:         else
 71:         {
 72:             while ( !(scr2.back) && !(finalAutonChoice) )
 73:             //if user selects a program with an auton
 74:             {
 75:                 autonConfig cnfg = scr2.getOptions( auton ); //get config options
 76:
 77:                 if ( !(scr2.back) ) //if user does not want to go back from screen 2
 78:                 {
 79:
 80:                     scr3.getConfirmation( auton ); //gets confirmation from user
 81:                     if ( scr3.confirm )
 82:                     {
 83:                         finalAutonChoice = auton;
 84:                         //scr4.updateLabels(interval, finalAutonChoice);
 85:                     }
 86:
 87:
 88:                 }
 89:                 else
 90:                 {
 91:                     break;
 92:                 }
 93:
 94:
 95:             }
 96:         }
 97:
 98:     }
 99:
100:
101:
102:     return finalAutonChoice;
103: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/DriverControl/DriverControlLCD.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: add actual content instead of blank screen
7:   *
8:   * contains the lcd screen used during driver control
9:   *
10:  */
11: #ifndef __DRIVERCONTROLLCD_HPP__
12: #define __DRIVERCONTROLLCD_HPP__
13:
14: #include "../../../../include/main.h"
15:
16: #include "../Styles.hpp"
17:
18:
19: /**
20:  * @see: ../Styles.hpp
21:  *
22:  * contains lcd to be used during driver control
23:  */
24: class DriverControlLCD : private Styles
25: {
26:     private:
27:         lv_obj_t *screen;
28:
29:         //labels
30:         lv_obj_t *title_label;
31:
32:
33:     public:
34:         DriverControlLCD();
35:         ~DriverControlLCD();
36:
37:         static int auton;
38:
39:         /**
40:          * @return: None
41:          *
42:          * TODO: add actual content to be updated
43:          *
44:          * function to be used to update the gui to keep data relevant
45:          */
46:         void updateLabels();
47:
48:
49: };
50:
51: #endif
```

```
 1: /**
 2:  * @file: ./RobotCode/src/lcdCode/DriverControl/DriverControlLCD.cpp
 3:  * @author: Aiden Carney
 4:  * @reviewed_on: 10/15/2019
 5:  * @reviewed_by: Aiden Carney
 6:  *
 7:  * @see: DriverControlLCD.hpp
 8:  *
 9:  * contains methods for driver control lcd
10:  */
11:
12: #include "../../../../include/main.h"
13: #include "../../../../include/api.h"
14:
15: #include "../../../Autons.hpp"
16: #include "DriverControlLCD.hpp"
17:
18:
19: int DriverControlLCD::auton = 0;
20:
21: DriverControlLCD::DriverControlLCD()
22: {
23:     screen = lv_obj_create(NULL, NULL);
24:     lv_obj_set_style(screen, &gray);
25: }
26:
27:
28:
29: DriverControlLCD::~DriverControlLCD()
30: {
31:     lv_obj_del(screen);
32: }
33:
34:
35:
36:
37: /**
38:  * updates colors and borders during driver control
39:  * keeps data relevent
40:  */
41: void DriverControlLCD::updateLabels()
42: {
43:     Autons auton_data;
44:
45:     //set background
46:     std::string color = auton_data.AUTONOMOUS_COLORS.at(auton);
47:     if (color == "blue")
48:     {
49:         gray.body.border.color = BLUE_BORDER;
50:     }
51:     else if (color == "red")
52:     {
53:         gray.body.border.color = RED_BORDER;
54:     }
55:     else
56:     {
57:         gray.body.border.color = BG;
58:     }
59:
60:     lv_scr_load(screen);
61:
62:     while ( 1 )
63:     {
64:         lv_obj_set_style(screen, &gray);
65:     }
66: }
```

```
 1: /**
 2:  * @file: ./RobotCode/src/lcdCode/DriverControl/LCDTask.hpp
 3:  * @author: Aiden Carney
 4:  * @reviewed_on: 10/15/2019
 5:  * @reviewed_by: Aiden Carney
 6:  * TODO: deprecate or update, lcd will not run on pros task which is what this is designed to do
 7:  *
 8:  * function to run task for lcd during driver control to update
 9:  *
10:  */
11:
12: #ifndef __LCDTASK_HPP__
13: #define __LCDTASK_HPP__
14:
15: #include "../../../../include/main.h"
16: #include "../../../../include/api.h"
17:
18: #include "DriverControlLCD.hpp"
19:
20: /**
21:  * @param: void* -> not used, required to implement wiht pros task api
22:  * @return: None
23:  *
24:  * @see: DriverControlLCD.hpp
25:  *
26:  * task that updates lcd during driver control
27:  *
28:  */
29: void lcd_task(void*);
30:
31: #endif
```

```
 1: /**
 2:  * @file: ./RobotCode/src/lcdCode/DriverControl/LCDTask.cpp
 3:  * @author: Aiden Carney
 4:  * @reviewed_on: 10/15/2019
 5:  * @reviewed_by: Aiden Carney
 6:  *
 7:  * @see: LCDTask.hpp
 8:  *
 9:  * description of contents line 1
10:  * description of contents line 2
11:  * description of contents line 3
12:  *
13:  */
14:
15: #include "../../../../include/main.h"
16: #include "../../../../include/api.h"
17:
18: #include "DriverControlLCD.hpp"
19: #include "LCDTask.hpp"
20:
21:
22: /**
23:  * runs loop to update lcd in a task
24:  */
25: void lcd_task(void*)
26: {
27:     DriverControlLCD mainLCD;
28:
29:     while ( 1 )
30:     {
31:         mainLCD.updateLabels();
32:         pros::delay(100); //high number so that the majority of time spent
33:                           //on this thread will be low so that time can be used
34:                           //for more useful things like recording
35:         std::cout << lv_scr_act() << "\n";
36:         pros::delay(100);
37:     }
38: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/AutonSelecton/ActionsScreen.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: add actual content for when actions are decided on
7:   *
8:   * does nothing
9:   *
10:  */
11:
12: #ifndef __ACTIONSSCREEN_HPP__
13: #define __ACTIONSSCREEN_HPP__
14:
15:
16: #include "../include/main.h"
17:
18:
19:
20:
21: #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/AutonSelecton/ActionsScreen.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: ActionsScreen.hpp
8:   *
9:   * does nothing
10:  *
11:  */
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/AutonSelecton/OptionsScreen.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: add correct options when they are decided on, deprecate static options to reduce coupling
7:   *
8:   * contains class with methods to decide on options for auton
9:   *
10:  */
11:
12:  #ifndef __OPTIONSSCREEN_HPP__
13:  #define __OPTIONSSCREEN_HPP__
14:
15:
16:  #include "../../../../include/main.h"
17:
18:  #include "../../../Autons.hpp"
19:  #include "../Styles.hpp"
20:
21:  /**
22:   * @see: ../Styles.hpp
23:   * @see: ./AutonSelection
24:   * @see: ../gui.hpp
25:   *
26:   * contains methods to get options for auton period
27:   */
28:  class OptionsScreen : private Styles
29:  {
30:      private:
31:          //screen
32:          lv_obj_t *options_screen;
33:
34:          //labels
35:          lv_obj_t *title_label;
36:
37:          lv_obj_t *sw_use_hardcoded_label;
38:          lv_obj_t *sw_gyro_turn_label;
39:          lv_obj_t *sw_accelleration_ctrl_label;
40:          lv_obj_t *sw_check_motor_tmp_label;
41:          lv_obj_t *sw_use_previous_macros_label;
42:          lv_obj_t *sw_record_label;
43:
44:          //buttons
45:          lv_obj_t *btn_confirm;
46:          lv_obj_t *btn_back;
47:
48:          //button labels
49:          lv_obj_t *btn_confirm_label;
50:          lv_obj_t *btn_back_label;
51:
52:
53:          //switches
54:          lv_obj_t *sw_use_hardcoded;
55:          lv_obj_t *sw_gyro_turn;
56:          lv_obj_t *sw_accelleration_ctrl;
57:          lv_obj_t *sw_check_motor_tmp;
58:          lv_obj_t *sw_use_previous_macros;
59:          lv_obj_t *sw_record;
60:
61:
62:      public:
63:          OptionsScreen();
64:          ~OptionsScreen();
65:
66:          static autonConfig cnfg;
67:          static bool nextScreen;
68:          static bool back;
69:
70:
71:          //button functions
72:
73:          /**
74:           * @param: lv_obj_t* btn -> button that called the funtion
75:           * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
76:           *
77:           * button callback function used to set variable so that gui continues
78:           * to the next stage
79:           */
80:          static lv_res_t btn_confirm_action(lv_obj_t *btn);
81:
82:          /**
83:           * @param: lv_obj_t* btn -> button that called the funtion
84:           * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
85:           *
86:           * button callback function used to set variable so that gui goes back to
87:           * the previous stage
88:           */
89:          static lv_res_t btn_back_action(lv_obj_t *btn);
90:
91:
92:          //switch functions
93:
94:          /**
95:           * @param: lv_obj_t* sw -> switch object that was selected
96:           * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
97:           * TODO: merge with other functions to condense code and make it more expandable
98:           *
99:           * sets configuration option for using a compiled auton vs auton written on sd card
100:          */
101:         static lv_res_t sw_use_hardcoded_action(lv_obj_t *sw);
102:
103:         /**
104:          * @param: lv_obj_t* sw -> switch object that was selected
105:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
106:          * TODO: merge with other functions to condense code and make it more expandable
107:          *
108:          * sets configuration option for using gyro turns in auton
109:          */
110:         static lv_res_t sw_gyro_turn_action(lv_obj_t *sw);
111:
112:         /**
113:          * @param: lv_obj_t* sw -> switch object that was selected
```

```
114:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
115:         * TODO: merge with other functions to condense code and make it more expandable
116:         *
117:         * sets configuration option for using acceleration control code
118:         */
119:        static lv_res_t sw_accelleration_ctrl_action(lv_obj_t *sw);
120:
121:        /**
122:         * @param: lv_obj_t* sw -> switch object that was selected
123:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
124:         * TODO: merge with other functions to condense code and make it more expandable
125:         *
126:         * sets configuration option for limiting motor output based on temperature during the match
127:         */
128:        static lv_res_t sw_check_motor_tmp_action(lv_obj_t *sw);
129:
130:        /**
131:         * @param: lv_obj_t* sw -> switch object that was selected
132:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
133:         * TODO: merge with other functions to condense code and make it more expandable
134:         *
135:         * sets configuration option for allowing the use of previously recorded macros
136:         */
137:        static lv_res_t sw_use_previous_macros_action(lv_obj_t *sw);
138:
139:        /**
140:         * @param: lv_obj_t* sw -> switch object that was selected
141:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
142:         * TODO: merge with other functions to condense code and make it more expandable
143:         *
144:         * sets configuration option for recording match in the macro format
145:         */
146:        static lv_res_t sw_record_action(lv_obj_t *sw);
147:
148:
149:
150:        /**
151:         * @param: int auton -> number of auton selected, used to set color background of lcd
152:         * @return: autonConfig -> configuration struct with options based on how the switches were set
153:         *
154:         * @see: ../Structs.hpp
155:         * @see: ../Gui.hpp
156:         *
157:         * allows user to interact with the switches to set configuration options
158:         * user can choose to go back or continue with the options selected
159:         */
160:        autonConfig getOptions( int auton );
161:
162:    };
163:
164:
165:    #endif
```

```
  1: /**
  2:  * @file: ./RobotCode/src/lcdCode/AutonSelecton/OptionsScreen.cpp
  3:  * @author: Aiden Carney
  4:  * @reviewed_on: 10/15/2019
  5:  * @reviewed_by: Aiden Carney
  6:  *
  7:  * @see: OptionsScreen.hpp
  8:  *
  9:  * contains class with methods for getting a configuration struct for how
 10:  * auton will be run
 11:  */
 12:
 13: #include <unordered_map>
 14:
 15: #include "../../../../include/main.h"
 16: #include "../../../../include/api.h"
 17:
 18: #include "../../../Autons.hpp"
 19: #include "../../controller/controller.hpp"
 20: #include "OptionsScreen.hpp"
 21:
 22:
 23: autonConfig OptionsScreen::cnfg;
 24: bool OptionsScreen::nextScreen = false;
 25: bool OptionsScreen::back = false;
 26:
 27:
 28:
 29: OptionsScreen::OptionsScreen()
 30: {
 31:     nextScreen = false;
 32:     back = false;
 33:
 34:     options_screen = lv_obj_create(NULL, NULL);
 35:
 36: //use hard coded autonomous
 37:     //switch
 38:     sw_use_hardcoded = lv_sw_create(options_screen, NULL); //switch template
 39:     lv_sw_set_style(sw_use_hardcoded, LV_SW_STYLE_BG, &sw_bg);
 40:     lv_sw_set_style(sw_use_hardcoded, LV_SW_STYLE_INDIC, &sw_indic);
 41:     lv_sw_set_style(sw_use_hardcoded, LV_SW_STYLE_KNOB_ON, &sw_toggled);
 42:     lv_sw_set_style(sw_use_hardcoded, LV_SW_STYLE_KNOB_OFF, &sw_off);
 43:
 44:     lv_sw_set_action(sw_use_hardcoded, sw_use_hardcoded_action); //map action
 45:     lv_obj_set_width(sw_use_hardcoded, 40); //witdth
 46:     lv_obj_set_height(sw_use_hardcoded, 20); //height
 47:
 48:     //label
 49:     sw_use_hardcoded_label = lv_label_create(options_screen, NULL);
 50:     lv_label_set_style(sw_use_hardcoded_label, &heading_text);
 51:     lv_obj_set_width(sw_use_hardcoded_label, 10);
 52:     lv_obj_set_height(sw_use_hardcoded_label, 20);
 53:     lv_label_set_align(sw_use_hardcoded_label, LV_LABEL_ALIGN_LEFT);
 54:     lv_label_set_text(sw_use_hardcoded_label, "Use Hardcoded Auton");
 55:
 56:
 57:
 58: //gyro turns
 59:     //switch
 60:     sw_gyro_turn = lv_sw_create(options_screen, sw_use_hardcoded);
 61:     lv_sw_set_action(sw_gyro_turn, sw_gyro_turn_action);
 62:     lv_obj_set_width(sw_gyro_turn, 40);
 63:     lv_obj_set_height(sw_gyro_turn, 20);
 64:
 65:     //label
 66:     sw_gyro_turn_label = lv_label_create(options_screen, NULL);
 67:     lv_label_set_style(sw_gyro_turn_label, &heading_text);
 68:     lv_obj_set_width(sw_gyro_turn_label, 300);
 69:     lv_obj_set_height(sw_gyro_turn_label, 20);
 70:     lv_label_set_align(sw_gyro_turn_label, LV_LABEL_ALIGN_LEFT);
 71:     lv_label_set_text(sw_gyro_turn_label, "Use Gyro Turns");
 72:
 73:
 74: //acceleration control
 75:     //switch
 76:     sw_accelleration_ctrl = lv_sw_create(options_screen, sw_use_hardcoded);
 77:     lv_sw_set_action(sw_accelleration_ctrl, sw_accelleration_ctrl_action);
 78:     lv_obj_set_width(sw_accelleration_ctrl, 40);
 79:     lv_obj_set_height(sw_accelleration_ctrl, 20);
 80:
 81:     //label
 82:     sw_accelleration_ctrl_label = lv_label_create(options_screen, NULL);
 83:     lv_label_set_style(sw_accelleration_ctrl_label, &heading_text);
 84:     lv_obj_set_width(sw_accelleration_ctrl_label, 300);
 85:     lv_obj_set_height(sw_accelleration_ctrl_label, 20);
 86:     lv_label_set_align(sw_accelleration_ctrl_label, LV_LABEL_ALIGN_LEFT);
 87:     lv_label_set_text(sw_accelleration_ctrl_label, "Use Acceleration Control");
 88:
 89:
 90: //check motor temp
 91:     //switch
 92:     sw_check_motor_tmp = lv_sw_create(options_screen, sw_use_hardcoded);
 93:     lv_sw_set_action(sw_check_motor_tmp, sw_check_motor_tmp_action);
 94:     lv_obj_set_width(sw_check_motor_tmp, 40);
 95:     lv_obj_set_height(sw_check_motor_tmp, 20);
 96:
 97:     //label
 98:     sw_check_motor_tmp_label = lv_label_create(options_screen, NULL);
 99:     lv_label_set_style(sw_check_motor_tmp_label, &heading_text);
100:     lv_obj_set_width(sw_check_motor_tmp_label, 300);
101:     lv_obj_set_height(sw_check_motor_tmp_label, 20);
102:     lv_label_set_align(sw_check_motor_tmp_label, LV_LABEL_ALIGN_LEFT);
103:     lv_label_set_text(sw_check_motor_tmp_label, "Limit Motor Temp");
104:
105:
106: //use previous macros
107:     //switch
108:     sw_use_previous_macros = lv_sw_create(options_screen, sw_use_hardcoded);
109:     lv_sw_set_action(sw_use_previous_macros, sw_use_previous_macros_action);
110:     lv_obj_set_width(sw_use_previous_macros, 40);
111:     lv_obj_set_height(sw_use_previous_macros, 20);
112:
113:     //label
```

```
114:     sw_use_previous_macros_label = lv_label_create(options_screen, NULL);
115:     lv_label_set_style(sw_use_previous_macros_label, &heading_text);
116:     lv_obj_set_width(sw_use_previous_macros_label, 300);
117:     lv_obj_set_height(sw_use_previous_macros_label, 20);
118:     lv_label_set_align(sw_use_previous_macros_label, LV_LABEL_ALIGN_LEFT);
119:     lv_label_set_text(sw_use_previous_macros_label, "Use Previously Recorded Macros");
120:
121:
122:   //record
123:     //switch
124:     sw_record = lv_sw_create(options_screen, sw_use_hardcoded);
125:     lv_sw_set_action(sw_record, sw_record_action);
126:     lv_obj_set_width(sw_record, 40);
127:     lv_obj_set_height(sw_record, 20);
128:
129:     //label
130:     sw_record_label = lv_label_create(options_screen, NULL);
131:     lv_label_set_style(sw_record_label, &heading_text);
132:     lv_obj_set_width(sw_record_label, 300);
133:     lv_obj_set_height(sw_record_label, 20);
134:     lv_label_set_align(sw_record_label, LV_LABEL_ALIGN_LEFT);
135:     lv_label_set_text(sw_record_label, "Record Motor Movements");
136:
137:
138:   //confirm button
139:     //button
140:     btn_confirm = lv_btn_create(options_screen, NULL);
141:     lv_btn_set_style(btn_confirm, LV_BTN_STYLE_REL, &toggle_btn_released);
142:     lv_btn_set_style(btn_confirm, LV_BTN_STYLE_PR, &toggle_btn_pressed);
143:     lv_btn_set_action(btn_confirm, LV_BTN_ACTION_CLICK, btn_confirm_action);
144:     lv_obj_set_width(btn_confirm, 300);
145:     lv_obj_set_height(btn_confirm, 25);
146:
147:     //label
148:     btn_confirm_label = lv_label_create(btn_confirm, NULL);
149:     lv_obj_set_style(btn_confirm_label, &heading_text);
150:     lv_label_set_text(btn_confirm_label, "Confirm");
151:
152:   //back button
153:     //button
154:     btn_back = lv_btn_create(options_screen, NULL);
155:     lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
156:     lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
157:     lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
158:     lv_obj_set_width(btn_back, 50);
159:     lv_obj_set_height(btn_back, 25);
160:
161:     //label
162:     btn_back_label = lv_label_create(btn_back, NULL);
163:     lv_obj_set_style(btn_back_label, &heading_text);
164:     lv_label_set_text(btn_back_label, "Back");
165:
166:
167:   //title label
168:     title_label = lv_label_create(options_screen, NULL);
169:     lv_obj_set_style(title_label, &heading_text);
170:     lv_obj_set_width(title_label, 300);
171:     lv_obj_set_height(title_label, 20);
172:     lv_label_set_align(title_label, LV_LABEL_ALIGN_LEFT);
173:     lv_label_set_text(title_label, "Auton");
174:
175:   //set postition of widgets
176:     lv_obj_set_pos(sw_use_hardcoded, 400, 40);
177:     lv_obj_set_pos(sw_gyro_turn, 400, 65);
178:     lv_obj_set_pos(sw_accelleration_ctrl, 400, 90);
179:     lv_obj_set_pos(sw_check_motor_tmp, 400, 115);
180:     lv_obj_set_pos(sw_use_previous_macros, 400, 140);
181:     lv_obj_set_pos(sw_record, 400, 165);
182:
183:     lv_obj_set_pos(sw_use_hardcoded_label, 20, 40);
184:     lv_obj_set_pos(sw_gyro_turn_label, 20, 65);
185:     lv_obj_set_pos(sw_accelleration_ctrl_label, 20, 90);
186:     lv_obj_set_pos(sw_check_motor_tmp_label, 20, 115);
187:     lv_obj_set_pos(sw_use_previous_macros_label, 20, 140);
188:     lv_obj_set_pos(sw_record_label, 20, 165);
189:
190:     lv_obj_set_pos(btn_back, 40, 200);
191:     lv_obj_set_pos(btn_confirm, 100, 200);
192:     lv_obj_set_pos(title_label, 210, 20);
193:   }
194:
195:
196:   OptionsScreen::~OptionsScreen()
197:   {
198:     lv_obj_del(sw_use_hardcoded_label);
199:     lv_obj_del(sw_gyro_turn_label);
200:     lv_obj_del(sw_accelleration_ctrl_label);
201:     lv_obj_del(sw_check_motor_tmp_label);
202:     lv_obj_del(sw_use_previous_macros_label);
203:     lv_obj_del(sw_record_label);
204:
205:     lv_obj_del(sw_use_hardcoded);
206:     lv_obj_del(sw_gyro_turn);
207:     lv_obj_del(sw_accelleration_ctrl);
208:     lv_obj_del(sw_check_motor_tmp);
209:     lv_obj_del(sw_use_previous_macros);
210:     lv_obj_del(sw_record);
211:
212:     lv_obj_del(btn_back_label);
213:     lv_obj_del(btn_confirm_label);
214:     lv_obj_del(title_label);
215:
216:     lv_obj_del(btn_back);
217:     lv_obj_del(btn_confirm);
218:
219:     lv_obj_del(options_screen);
220:
221:   }
222:
223:
224:
225:   /**
226:    * sets nextScreen so that main loop will break and go to next stage
```

```
227:  */
228:  lv_res_t OptionsScreen::btn_confirm_action(lv_obj_t *btn)
229:  {
230:     nextScreen = true;
231:     back = false;
232:
233:     return LV_RES_OK;
234:  }
235:
236:  /**
237:   * sets nextScreen so that main loop will break and go to the previous stage
238:   */
239:  lv_res_t OptionsScreen::btn_back_action(lv_obj_t *btn)
240:  {
241:     nextScreen = true;
242:     back = true;
243:
244:     return LV_RES_OK;
245:  }
246:
247:
248:
249:
250:  /**
251:   * sets or clears config option for using hard coded autons based on the
252:   * switches previous state
253:   */
254:  lv_res_t OptionsScreen::sw_use_hardcoded_action(lv_obj_t *sw)
255:  {
256:     cnfg.use_hardcoded = !(cnfg.use_hardcoded);
257:     return LV_RES_OK;
258:  }
259:
260:  /**
261:   * sets or clears config option for using gyro turns based on the
262:   * switches previous state
263:   */
264:  lv_res_t OptionsScreen::sw_gyro_turn_action(lv_obj_t *sw)
265:  {
266:     cnfg.gyro_turn = !(cnfg.gyro_turn);
267:     return LV_RES_OK;
268:  }
269:
270:
271:  /**
272:   * sets or clears config option for using acceleration control code based on the
273:   * switches previous state
274:   */
275:  lv_res_t OptionsScreen::sw_accelleration_ctrl_action(lv_obj_t *sw)
276:  {
277:     cnfg.accelleration_ctrl = !(cnfg.accelleration_ctrl);
278:     return LV_RES_OK;
279:  }
280:
281:
282:  /**
283:   * sets or clears config option for limiting motor output based on the
284:   * switches previous state
285:   */
286:  lv_res_t OptionsScreen::sw_check_motor_tmp_action(lv_obj_t *sw)
287:  {
288:     cnfg.check_motor_tmp = !(cnfg.check_motor_tmp);
289:     return LV_RES_OK;
290:  }
291:
292:
293:  /**
294:   * sets or clears config option for allowing use of previously recorded macros based on the
295:   * switches previous state
296:   */
297:  lv_res_t OptionsScreen::sw_use_previous_macros_action(lv_obj_t *sw)
298:  {
299:     cnfg.use_previous_macros = !(cnfg.use_previous_macros);
300:     return LV_RES_OK;
301:  }
302:
303:
304:  /**
305:   * sets or clears config option for recorded the match as a macro based on the
306:   * switches previous state
307:   */
308:  lv_res_t OptionsScreen::sw_record_action(lv_obj_t *sw)
309:  {
310:     cnfg.record = !(cnfg.record);
311:     return LV_RES_OK;
312:  }
313:
314:
315:
316:  /**
317:   * runs loop where user can set auton configuration options with digital switches
318:   * loop breaks when user clicks the back or continue button
319:   * if user click the back button then the back flag is set
320:   */
321:  autonConfig OptionsScreen::getOptions( int auton )
322:  {
323:     Controller controllers;
324:     Autons auton_data;
325:
326:     lv_sw_off(sw_use_hardcoded); //reset switches and values
327:     lv_sw_on(sw_gyro_turn);
328:     lv_sw_on(sw_accelleration_ctrl);
329:     lv_sw_off(sw_check_motor_tmp);
330:     lv_sw_on(sw_use_previous_macros);
331:     lv_sw_off(sw_record);
332:
333:     cnfg.use_hardcoded = 0;
334:     cnfg.gyro_turn = 1;
335:     cnfg.accelleration_ctrl = 1;
336:     cnfg.check_motor_tmp = 0;
337:     cnfg.use_previous_macros = 1;
338:     cnfg.record = 0;
339:
```

```
340:
341:    lv_label_set_text(title_label, auton_data.AUTONOMOUS_NAMES.at(auton));
342:
343:    //load screen
344:    lv_scr_load(options_screen);
345:
346:    //set color of border
347:    std::string color = auton_data.AUTONOMOUS_COLORS.at(auton);
348:    if (color == "blue")
349:    {
350:       gray.body.border.color = BLUE_BORDER;
351:    }
352:    else if (color == "red")
353:    {
354:       gray.body.border.color = RED_BORDER;
355:    }
356:    else
357:    {
358:       gray.body.border.color = BG;
359:    }
360:
361:    lv_obj_set_style(options_screen, &gray);
362:
363:
364:    back = false;
365:    nextScreen = false;
366:
367:    pros::delay( 100 );  //add delay so that previous button clicks do not register
368:
369:    while ( !(nextScreen) )
370:    {
371:       //allow controller to press the buttons as well
372:       if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_A) )
373:       {
374:          btn_confirm_action( NULL );
375:          pros::delay(100);
376:       }
377:       else if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_B) )
378:       {
379:          btn_back_action( NULL );
380:          pros::delay(100);
381:       }
382:
383:       pros::delay(20);
384:    }
385:
386:    return cnfg;
387:
388: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/AutonSelecton/PrepScreen.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: add actual preparation steps text
7:   * TODO: decouple initilization string, make it more configurable
8:   *
9:   * contains class with methods for showing the things that will occur
10:  * before the auton is run
11:  */
12:
13: #ifndef __PREP_SCREEN__
14: #define __PREP_SCREEN__
15:
16:
17: #include "../include/main.h"
18:
19: #include "../Styles.hpp"
20: #include "OptionsScreen.hpp"
21:
22:
23: /**
24:  * @see: ../Styles.hpp
25:  * @see: ../gui.hpp
26:  *
27:  * final confirmation steps for auton
28:  * shows the initialization that will occur after the user clicks continue
29:  */
30: class PrepScreen : private Styles
31: {
32:     private:
33:         //screen
34:         lv_obj_t *prep_screen;
35:
36:         //labels
37:         lv_obj_t *title_label;
38:
39:         lv_obj_t *actions_label;
40:
41:         //buttons
42:         lv_obj_t *btn_confirm;
43:         lv_obj_t *btn_back;
44:
45:         //button labels
46:         lv_obj_t *btn_confirm_label;
47:         lv_obj_t *btn_back_label;
48:
49:
50:     public:
51:         PrepScreen();
52:         ~PrepScreen();
53:
54:         static bool nextScreen;
55:         static bool confirm;
56:
57:         //button functions
58:
59:         /**
60:          * @param: lv_obj_t* btn -> button that called the funtion
61:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
62:          *
63:          * button callback function used to set variable so that gui continues
64:          * to the next stage
65:          */
66:         static lv_res_t btn_confirm_action(lv_obj_t *btn);
67:
68:
69:         /**
70:          * @param: lv_obj_t* btn -> button that called the funtion
71:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
72:          *
73:          * button callback function used to set variable so that gui goes
74:          * to the previous stage
75:          */
76:         static lv_res_t btn_back_action(lv_obj_t *btn);
77:
78:
79:         /**
80:          * @param: int auton -> auton number selected, used to set border color of gui based on side of color the auton is run on
81:          * @return: None
82:          *
83:          * @see: ../Structs.hpp
84:          * @see: ../Styles.hpp
85:          *
86:          * gunction used to get confirmation from user to continue to next stage
87:          * of the selection process
88:          */
89:         void getConfirmation( int auton );
90:
91: };
92: #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/AutonSelecton/PrepScreen.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: PrepScreen.hpp
8:   *
9:   * contains class methods seeking confirmation from user
10:  */
11:
12:  #include <sstream>
13:  #include <string>
14:
15:  #include "../../../../include/main.h"
16:  #include "../../../../include/api.h"
17:
18:  #include "../../../Autons.hpp"
19:  #include "../../controller/controller.hpp"
20:  #include "PrepScreen.hpp"
21:
22:
23:  bool PrepScreen::nextScreen = false;
24:  bool PrepScreen::confirm = false;
25:
26:
27:
28:  PrepScreen::PrepScreen()
29:  {
30:      nextScreen = false;
31:      confirm = false;
32:
33:      prep_screen = lv_obj_create(NULL, NULL);
34:
35:  //actions_label
36:      actions_label = lv_label_create(prep_screen, NULL);
37:      lv_obj_set_style(actions_label, &heading_text);
38:      lv_obj_set_width(actions_label, 300);
39:      lv_obj_set_height(actions_label, 160);
40:      lv_label_set_align(actions_label, LV_LABEL_ALIGN_LEFT);
41:      lv_label_set_text(actions_label, "actions");
42:
43:
44:
45:  //confirm button
46:      //button
47:      btn_confirm = lv_btn_create(prep_screen, NULL);
48:      lv_btn_set_style(btn_confirm, LV_BTN_STYLE_REL, &toggle_btn_released);
49:      lv_btn_set_style(btn_confirm, LV_BTN_STYLE_PR, &toggle_btn_pressed);
50:      lv_btn_set_action(btn_confirm, LV_BTN_ACTION_CLICK, btn_confirm_action);
51:      lv_obj_set_width(btn_confirm, 300);
52:      lv_obj_set_height(btn_confirm, 25);
53:
54:      //label
55:      btn_confirm_label = lv_label_create(btn_confirm, NULL);
56:      lv_obj_set_style(btn_confirm_label, &heading_text);
57:      lv_label_set_text(btn_confirm_label, "Confirm");
58:
59:  //back button
60:      //button
61:      btn_back = lv_btn_create(prep_screen, NULL);
62:      lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
63:      lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
64:      lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
65:      lv_obj_set_width(btn_back, 50);
66:      lv_obj_set_height(btn_back, 25);
67:
68:      //label
69:      btn_back_label = lv_label_create(btn_back, NULL);
70:      lv_obj_set_style(btn_back_label, &heading_text);
71:      lv_label_set_text(btn_back_label, "Back");
72:
73:
74:  //title label
75:      title_label = lv_label_create(prep_screen, NULL);
76:      lv_obj_set_style(title_label, &heading_text);
77:      lv_obj_set_width(title_label, 300);
78:      lv_obj_set_height(title_label, 20);
79:      lv_label_set_align(title_label, LV_LABEL_ALIGN_LEFT);
80:      lv_label_set_text(title_label, "Auton");
81:
82:
83:      lv_obj_set_pos(btn_back, 40, 200);
84:      lv_obj_set_pos(btn_confirm, 100, 200);
85:      lv_obj_set_pos(title_label, 210, 20);
86:      lv_obj_set_pos(actions_label, 40, 50);
87:  }
88:
89:
90:
91:
92:  PrepScreen::~PrepScreen()
93:  {
94:      lv_obj_del(btn_back_label);
95:      lv_obj_del(btn_confirm_label);
96:      lv_obj_del(title_label);
97:
98:      lv_obj_del(btn_back);
99:      lv_obj_del(btn_confirm);
100:
101:     lv_obj_del(prep_screen);
102: }
103:
104:
105:
106:
107: /**
108:  * sets nextScreen so that main loop will break and go to next stage
109:  */
110: lv_res_t PrepScreen::btn_confirm_action(lv_obj_t *btn)
111: {
112:     nextScreen = true;
113:     confirm = true;
```

```
114:
115:     return LV_RES_OK;
116:  }
117:
118:  /**
119:   * sets nextScreen so that main loop will break and go to the previous stage
120:   */
121:  lv_res_t PrepScreen::btn_back_action(lv_obj_t *btn)
122:  {
123:     nextScreen = true;
124:     confirm = false;
125:
126:     return LV_RES_OK;
127:  }
128:
129:
130:
131:  /**
132:   * runs loop where user can see what operations will be performed
133:   * loop breaks when user clicks the back or continue button
134:   * if user click the back button then the back flag is set
135:   */
136:  void PrepScreen::getConfirmation( int auton )
137:  {
138:     Controller controllers;
139:     Autons auton_data;
140:
141:     lv_label_set_text(title_label, auton_data.AUTONOMOUS_NAMES.at(auton));
142:
143:     //load screen
144:     lv_scr_load(prep_screen);
145:
146:     //set color of border
147:     std::string color = auton_data.AUTONOMOUS_COLORS.at(auton);
148:     if (color == "blue")
149:     {
150:        gray.body.border.color = BLUE_BORDER;
151:     }
152:     else if (color == "red")
153:     {
154:        gray.body.border.color = RED_BORDER;
155:     }
156:     else
157:     {
158:        gray.body.border.color = BG;
159:     }
160:
161:     lv_obj_set_style(prep_screen, &gray);
162:
163:     nextScreen = false;
164:
165:
166:     std::string label =
167:        "Initialize and Calibrate Gyro\n"
168:        "Initialize Other Sensors\n"
169:        "Initialize Motors\n"
170:        "Zero Motor Encoders\n"
171:        "Initialize Controllers\n";
172:     if ( OptionsScreen::cnfg.record )
173:     {
174:        label = label + "Start Recording Thread";
175:     }
176:
177:     //cast std::string to const char* and set text
178:     std::ostringstream text;
179:     text << label;
180:     lv_label_set_text(actions_label, text.str().c_str());
181:
182:     pros::delay( 100 );  //add delay so that button press from previous stage does not register
183:
184:     while ( !(nextScreen) )
185:     {
186:        if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_A) )
187:        {
188:           btn_confirm_action( NULL );
189:           pros::delay(200);
190:        }
191:        else if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_B) )
192:        {
193:           btn_back_action( NULL );
194:           pros::delay(200);
195:        }
196:        pros::delay(20);
197:     }
198:
199:  }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/AutonSelecton/SelectionScreen.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: add ability for controller to make selections
7:   *
8:   * contains first stage of auton selection--selecting the auton number
9:   *
10:  */
11:
12: #ifndef __SELECTIONSCREEN_HPP__
13: #define __SELECTIONSCREEN_HPP__
14:
15:
16: #include "../../../../include/main.h"
17:
18: #include "../Styles.hpp"
19:
20:
21:
22:
23: /**
24:  * @see: ../Styles.hpp
25:  *
26:  * contains methods for going through each auton and providing description
27:  * for user
28:  */
29: class SelectionScreen : private Styles
30: {
31:     private:
32:
33:         //labels
34:         lv_obj_t *title_label;
35:         lv_obj_t *description_label;
36:         lv_obj_t *auton_number_label;
37:
38:         //buttons
39:         lv_obj_t *btn_right;
40:         lv_obj_t *btn_left;
41:         lv_obj_t *btn_select;
42:
43:         //button labels
44:         lv_obj_t *btn_right_label;
45:         lv_obj_t *btn_left_label;
46:         lv_obj_t *btn_select_label;
47:
48:
49:     public:
50:         //screens
51:         static lv_obj_t *selection_screen;
52:         //variables
53:         static int auton_choice;
54:         static int final_choice;
55:         static bool update;
56:
57:         //button action functions
58:
59:         /**
60:          * @param: lv_obj_t* btn -> button that called the funtion
61:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
62:          *
63:          * button callback function used to set variable so that gui moves
64:          * autons to the right (increasing auton number by one and looping at end)
65:          */
66:         static lv_res_t btn_right_action(lv_obj_t *btn);
67:
68:
69:         /**
70:          * @param: lv_obj_t* btn -> button that called the funtion
71:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
72:          *
73:          * button callback function used to set variable so that gui moves
74:          * autons to the left (decreasing auton number by one and looping at end)
75:          */
76:         static lv_res_t btn_left_action(lv_obj_t *btn);
77:
78:
79:         /**
80:          * @param: lv_obj_t* btn -> button that called the funtion
81:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
82:          *
83:          * button callback function used to select the auton so that the gui continues
84:          * to the next stage
85:          */
86:         static lv_res_t btn_select_action(lv_obj_t *btn);
87:
88:         SelectionScreen();
89:         ~SelectionScreen();
90:
91:
92:
93:         /**
94:          * @return: None
95:          *
96:          * @see: int selectAuton()
97:          * @see: ../Structs.hpp
98:          *
99:          * sets the description, color, title, and auton number on the screen
100:         * used to update the auton selection based on the current number
101:         */
102:        void showSelection();
103:
104:
105:        /**
106:         * @param: int auton -> auton number to start the screen at
107:         * @return: int -> auton number that the screen was on when the user hit the select button
108:         *
109:         * loops through waiting for the auton number to change to update the screen
110:         */
111:        int selectAuton( int auton );
112:
113:
```

```
114:  };
115:
116:
117:
118:  #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/AutonSelecton/SelectionScreen.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: SelectionScreen.hpp
8:   *
9:   * contains methods for class that give user the ability to select an auton number
10:  */
11:
12:  #include <sstream>
13:  #include <string>
14:
15:  #include "../../../../include/main.h"
16:
17:  #include "../../../Autons.hpp"
18:  #include "SelectionScreen.hpp"
19:  #include "../../controller/controller.hpp"
20:
21:  //init static vars
22:  lv_obj_t *SelectionScreen::selection_screen;
23:  int SelectionScreen::auton_choice = 1;
24:  int SelectionScreen::final_choice = 0;
25:  bool SelectionScreen::update = false;
26:
27:
28:  //constructor
29:  SelectionScreen::SelectionScreen()
30:
31:  {
32:      auton_choice = 1;
33:      final_choice = 0;
34:      update = false;
35:
36:      selection_screen = lv_obj_create(NULL, NULL);
37:
38:
39:      //init buttons and labels
40:      title_label = lv_label_create(selection_screen, NULL);
41:      description_label = lv_label_create(selection_screen, NULL);
42:      auton_number_label = lv_label_create(selection_screen, NULL);
43:
44:      btn_right = lv_btn_create(selection_screen, NULL);
45:      btn_left = lv_btn_create(selection_screen, NULL);
46:      btn_select = lv_btn_create(selection_screen, NULL);
47:
48:      btn_right_label = lv_label_create(btn_right, NULL);
49:      btn_left_label = lv_label_create(btn_left, NULL);
50:      btn_select_label = lv_label_create(btn_select, NULL);
51:
52:
53:      //sets style for widgets
54:      lv_obj_set_style(selection_screen, &gray);
55:
56:      lv_btn_set_style(btn_right, LV_BTN_STYLE_REL, &toggle_btn_released);
57:      lv_btn_set_style(btn_left, LV_BTN_STYLE_REL, &toggle_btn_released);
58:      lv_btn_set_style(btn_select, LV_BTN_STYLE_REL, &toggle_btn_released);
59:
60:      lv_btn_set_style(btn_right, LV_BTN_STYLE_PR, &toggle_btn_pressed);
61:      lv_btn_set_style(btn_left, LV_BTN_STYLE_PR, &toggle_btn_pressed);
62:      lv_btn_set_style(btn_select, LV_BTN_STYLE_PR, &toggle_btn_pressed);
63:
64:      lv_obj_set_style(btn_right_label, &heading_text);
65:      lv_obj_set_style(btn_left_label, &heading_text);
66:      lv_obj_set_style(btn_select_label, &heading_text);
67:
68:      lv_label_set_style(title_label, &heading_text);
69:      lv_label_set_style(description_label, &heading_text);
70:      lv_label_set_style(auton_number_label, &heading_text);
71:
72:      lv_label_set_long_mode(description_label, LV_LABEL_LONG_BREAK);
73:      lv_label_set_align(auton_number_label, LV_LABEL_ALIGN_CENTER);
74:
75:      lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
76:      lv_label_set_align(description_label, LV_LABEL_ALIGN_CENTER);
77:
78:
79:      //set size of widgets
80:      lv_obj_set_width(btn_right, 80);
81:      lv_obj_set_width(btn_left, 80);
82:      lv_obj_set_width(btn_select, 120);
83:      lv_obj_set_width(auton_number_label, 40);
84:      lv_obj_set_width(title_label, 400);
85:      lv_obj_set_width(description_label, 400);
86:
87:      lv_obj_set_height(btn_right, 80);
88:      lv_obj_set_height(btn_left, 80);
89:      lv_obj_set_height(btn_select, 40);
90:      lv_obj_set_height(auton_number_label, 40);
91:      lv_obj_set_height(title_label, 30);
92:      lv_obj_set_height(description_label, 80);
93:
94:
95:      //set default text and move widgets to start location
96:      lv_label_set_text(btn_right_label, SYMBOL_RIGHT);
97:      lv_label_set_text(btn_left_label, SYMBOL_LEFT);
98:      lv_label_set_text(btn_select_label, "Select");
99:
100:     lv_obj_set_pos(btn_right, 390, 150);
101:     lv_obj_set_pos(btn_left, 10, 150);
102:     lv_obj_set_pos(btn_select, 180, 180);
103:     lv_obj_set_pos(auton_number_label, 440, 20);
104:     lv_obj_set_pos(title_label, 210, 20);
105:     lv_obj_set_pos(description_label, 40, 60);
106:
107:
108:     //set action for buttons
109:     lv_btn_set_action(btn_right, LV_BTN_ACTION_CLICK, btn_right_action);
110:     lv_btn_set_action(btn_left, LV_BTN_ACTION_CLICK, btn_left_action);
111:     lv_btn_set_action(btn_select, LV_BTN_ACTION_CLICK, btn_select_action);
112:
113: }
```

```
114:
115:
116: //destructor
117: SelectionScreen::~SelectionScreen()
118: {
119:     lv_obj_del(auton_number_label);
120:     lv_obj_del(title_label);
121:     lv_obj_del(description_label);
122:     lv_obj_del(btn_right);
123:     lv_obj_del(btn_left);
124:     lv_obj_del(btn_select);
125:
126:     lv_obj_del(selection_screen);
127: }
128:
129:
130:
131: //button action functions
132:
133: /**
134:  * called when left button is clicked
135:  * decrements auton_choice and loops it back in range if not in range
136:  */
137: lv_res_t SelectionScreen::btn_left_action(lv_obj_t *btn)
138: {
139:     Autons auton_data;
140:
141:     auton_choice -= 1;
142:     if (auton_choice < 1)
143:     {
144:         auton_choice = auton_data.AUTONOMOUS_NAMES.size();
145:     }
146:
147:     update = true;
148:
149:     return LV_RES_OK;
150: }
151:
152:
153: /**
154:  * called when left button is clicked
155:  * increments auton_choice and loops it back in range if not in range
156:  */
157: lv_res_t SelectionScreen::btn_right_action(lv_obj_t *btn)
158: {
159:     std::cout << "function called\n";
160:     Autons auton_data;
161:
162:     auton_choice += 1;
163:     if (auton_choice > auton_data.AUTONOMOUS_NAMES.size())
164:     {
165:         auton_choice = 1;
166:     }
167:
168:     update = true;
169:
170:     return LV_RES_OK;
171: }
172:
173:
174: /**
175:  * breaks main loop by setting the final auton choice so that gui continues
176:  */
177: lv_res_t SelectionScreen::btn_select_action(lv_obj_t *btn)
178: {
179:     final_choice = auton_choice;
180:     update = true;
181:
182:     return LV_RES_OK;
183: }
184:
185:
186:
187:
188: //other functions
189:
190: /**
191:  * updates background color by looking at std::unordered_map
192:  * updates auton number label
193:  * waits for there to be an update to be implemented by the buttons before exiting
194:  */
195: void SelectionScreen::showSelection()
196: {
197:     Controller controllers;
198:     Autons auton_data;
199:
200:     lv_label_set_text(title_label, auton_data.AUTONOMOUS_NAMES.at(auton_choice));
201:     lv_label_set_text(description_label, auton_data.AUTONOMOUS_DESCRIPTIONS.at(auton_choice));
202:
203:
204:     //get color
205:     std::string color = auton_data.AUTONOMOUS_COLORS.at(auton_choice);
206:     if (color == "blue")
207:     {
208:         gray.body.border.color = BLUE_BORDER;
209:     }
210:     else if (color == "red")
211:     {
212:         gray.body.border.color = RED_BORDER;
213:     }
214:     else
215:     {
216:         gray.body.border.color = BG;
217:     }
218:
219:     lv_obj_set_style(selection_screen, &gray); //update background
220:
221:     //cast int of auton choice to string
222:     std::string str_auton_choice;
223:     str_auton_choice = std::to_string(auton_choice);
224:     lv_label_set_text(auton_number_label, str_auton_choice.c_str());
225:
226:
```

```
227:     controllers.master.print( 0, 0, "                    ");
228:     pros::delay(50);
229:     controllers.master.print( 0, 0, auton_data.AUTONOMOUS_NAMES.at(auton_choice) );
230:
231:     while ( !(update) && !(final_choice) ) //waits for screen to change
232:                                 //so that time is not wasted
233:     {
234:       //allow controller to press the buttons as well
235:       if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_R1) )
236:       {
237:         btn_right_action( NULL );
238:         pros::delay(200);
239:       }
240:       else if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_L1) )
241:       {
242:         btn_left_action( NULL );
243:         pros::delay(200);
244:       }
245:       else if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_A) )
246:       {
247:         btn_select_action( NULL );
248:         pros::delay(200);
249:       }
250:       pros::delay(100);
251:     }
252:     update = false;
253:   }
254:
255:
256:
257:
258:   /**
259:    * waits in a loop for there to be an update to the gui implemented by
260:    * button callback functions
261:    * in the loop, the gui is updated until a final selection is made
262:    * everytime there is a change ie. when a button is clicked
263:    */
264:   int SelectionScreen::selectAuton( int auton )
265:   {
266:     auton_choice = auton;
267:     final_choice = 0;
268:     update = false;
269:
270:     //load screen
271:     lv_scr_load(selection_screen);
272:
273:     while ( !(final_choice) ) //waits for user to select an auton
274:     {                   //before going to next screen
275:       showSelection(); //showSelection contains delay
276:     }
277:
278:     gray.body.border.color = BG; //reset gray style
279:
280:     return final_choice;
281:
282:   }
```

```
 1: /**
 2:  * @file: ./RobotCode/src/lcdCode/Debug/Debug.hpp
 3:  * @author: Aiden Carney
 4:  * @reviewed_on: 10/15/2019
 5:  * @reviewed_by: Aiden Carney
 6:  * TODO: move starting of driver control task to this function because that task should be available during the debugger session
 7:  *
 8:  * gives user the option to visit debugger tabs by selecting an option from a button
 9:  * matrix
10:  */
11:
12: #ifndef __DEBUG_HPP__
13: #define __DEBUG_HPP__
14:
15: #include "BatteryDebug.hpp"
16: #include "ControllerDebug.hpp"
17: #include "FieldControlDebug.hpp"
18: #include "InternalMotorDebug.hpp"
19: #include "MotorsDebug.hpp"
20: #include "SensorsDebug.hpp"
21: #include "TitleScreen.hpp"
22: #include "Wiring.hpp"
23:
24:
25:
26: /**
27:  * @return: None
28:  *
29:  * @see: TitleScreen.hpp
30:  *
31:  * loads screens and switches the debugger option based on a what is clicked from
32:  * a button matrix
33:  */
34: void debug();
35:
36:
37: #endif
```

```
 1: /**
 2:  * @file: ./RobotCode/src/lcdCode/Debug/Debug.cpp
 3:  * @author: Aiden Carney
 4:  * @reviewed_on: 10/15/2019
 5:  * @reviewed_by: Aiden Carney
 6:  *
 7:  * @see: Debug.hpp
 8:  *
 9:  * contains function for selecting debug screen
10:  *
11:  */
12:
13: #include "../../../../include/main.h"
14: #include "../../../../include/api.h"
15:
16: #include "Debug.hpp"
17:
18:
19:
20: /**
21:  * loads all screens at beginning
22:  * when on titlescreen a tab number is selected and a switch statement is used
23:  * to let a tab take over
24:  */
25: void debug()
26: {
27:     bool cont = true;
28:
29:     TitleScreen dbg1;
30:     MotorsDebug dbgM;
31:     SensorsDebug dbgS;
32:     ControllerDebug dbgC;
33:     BatteryDebug dbgB;
34:     FieldControlDebug dbgF;
35:     Wiring dbgW;
36:     InternalMotorDebug dbgP;
37:
38:
39:     while ( cont )
40:     {
41:         dbg1.chooseOption();
42:         if ( dbg1.option == -1 ) //-1 means go back
43:         {
44:             cont = false;
45:             break;
46:         }
47:
48:         switch (dbg1.option) //go to selected debug screen
49:         {
50:             case 1:
51:                 dbgM.debug();
52:                 break;
53:
54:             case 2:
55:                 dbgS.debug();
56:                 break;
57:
58:             case 3:
59:                 dbgC.debug();
60:                 break;
61:
62:             case 4:
63:                 dbgB.debug();
64:                 break;
65:
66:             case 5:
67:                 dbgF.debug();
68:                 break;
69:
70:             case 6:
71:                 dbgW.debug();
72:                 break;
73:             case 7:
74:                 dbgP.debug();
75:                 break;
76:         }
77:
78:     }
79:
80: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/BatteryDebug.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/16/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * contains class for debugging the battery
8:   */
9:
10: #ifndef __BATTERYDEBUG_HPP__
11: #define __BATTERYDEBUG_HPP__
12:
13:
14: #include "../../../../include/main.h"
15:
16: #include "../Styles.hpp"
17:
18:
19: /**
20:  * @see: ../Styles.hpp
21:  *
22:  * contains methods that show user data about the battery
23:  */
24: class BatteryDebug : private Styles
25: {
26:     private:
27:         lv_obj_t *battery_screen;
28:         lv_obj_t *title_label;
29:
30:         lv_obj_t *labels_label;
31:         lv_obj_t *info_label;
32:
33:
34:         //back button
35:         lv_obj_t *btn_back;
36:         lv_obj_t *btn_back_label;
37:
38:         /**
39:          * @param: lv_obj_t* btn -> button that called the funtion
40:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
41:          *
42:          * button callback function used to set cont to false meaning the
43:          * user wants to go to the title screen
44:          */
45:         static lv_res_t btn_back_action(lv_obj_t *btn);
46:
47:     public:
48:         static bool cont;
49:
50:         BatteryDebug();
51:         ~BatteryDebug();
52:
53:
54:         /**
55:          * @return: None
56:          *
57:          * allows user to see information about the state of field control
58:          */
59:         void debug();
60:
61: };
62:
63:
64:
65:
66: #endif
```

```cpp
/**
 * @file: ./RobotCode/src/lcdCode/Debug/BatteryDebug.cpp
 * @author: Aiden Carney
 * @reviewed_on: 10/16/2019
 * @reviewed_by: Aiden Carney
 *
 * @see BatteryDebug.hpp
 *
 * contains implementation for class for debugging the battery
 */

#include "../../../../include/main.h"
#include "../../../../include/api.h"

#include "../Styles.hpp"
#include "BatteryDebug.hpp"


bool BatteryDebug::cont = true;

BatteryDebug::BatteryDebug()
{
    cont = true;

//screen
    battery_screen = lv_obj_create(NULL, NULL);
    lv_obj_set_style(battery_screen, &gray);

//init back button
    //button
    btn_back = lv_btn_create(battery_screen, NULL);
    lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
    lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
    lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
    lv_obj_set_width(btn_back, 75);
    lv_obj_set_height(btn_back, 25);

    //label
    btn_back_label = lv_label_create(btn_back, NULL);
    lv_obj_set_style(btn_back_label, &heading_text);
    lv_label_set_text(btn_back_label, "Back");

//init title label
    title_label = lv_label_create(battery_screen, NULL);
    lv_label_set_style(title_label, &heading_text);
    lv_obj_set_width(title_label, 440);
    lv_obj_set_height(title_label, 20);
    lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
    lv_label_set_text(title_label, "Field Control - Debug");

//init headers label
    labels_label = lv_label_create(battery_screen, NULL);
    lv_label_set_style(labels_label, &subheading_text);
    lv_obj_set_width(labels_label, 220);
    lv_obj_set_height(labels_label, 200);
    lv_label_set_align(labels_label, LV_LABEL_ALIGN_LEFT);

    std::string labels_label_text = (
        "battery percentage\n"
        "current\n"
        "voltage\n"
        "temperature"
    );

    lv_label_set_text(labels_label, labels_label_text.c_str());

//init values label
    info_label = lv_label_create(battery_screen, NULL);
    lv_label_set_style(info_label, &subheading_text);
    lv_obj_set_width(info_label, 220);
    lv_obj_set_height(info_label, 200);
    lv_label_set_align(info_label, LV_LABEL_ALIGN_LEFT);

    std::string info_label_text = (
        "None\n"
        "None\n"
        "None\n"
        "None"
    );

    lv_label_set_text(info_label, info_label_text.c_str());

//set positions
    lv_obj_set_pos(btn_back, 30, 210);

    lv_obj_align(title_label, battery_screen, LV_ALIGN_IN_TOP_MID, 0, 10);

    lv_obj_set_pos(labels_label, 20, 40);
    lv_obj_set_pos(info_label, 360, 40);


}


BatteryDebug::~BatteryDebug()
{
    lv_obj_del(battery_screen);
}



/**
 * sets cont to false to break main loop so main function returns
 */
lv_res_t BatteryDebug::btn_back_action(lv_obj_t *btn)
{
    cont = false;
    return LV_RES_OK;
}
```

```
114:  /**
115:   * main loop that updates the battery information
116:   */
117:  void BatteryDebug::debug()
118:  {
119:      cont = true;
120:
121:      lv_scr_load(battery_screen);
122:
123:      while ( cont )
124:      {
125:          std::string info_label_text = (
126:              std::to_string(pros::battery::get_capacity()) + "%\n"
127:              + std::to_string(pros::battery::get_current()) + "\n"
128:              + std::to_string(pros::battery::get_voltage()) + "\n"
129:              + std::to_string(pros::battery::get_temperature()) + "\n"
130:          );
131:
132:          lv_label_set_text(info_label, info_label_text.c_str());
133:
134:
135:          pros::delay(100);
136:      }
137:  }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/ControllerDebug.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/16/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO:
7:   *
8:   * contains classes to show data about controller on gui
9:   */
10:
11:  #ifndef __CONTROLLERDEBUG_HPP__
12:  #define __CONTROLLERDEBUG_HPP__
13:
14:  #include <unordered_map>
15:
16:  #include "../../../../include/main.h"
17:
18:  #include "../Styles.hpp"
19:
20:  //user defines
21:
22:  //sets size of container
23:  #define CONTROLLER_CONTAINER_WIDTH 440
24:  #define CONTROLLER_CONTAINER_HEIGHT 120
25:
26:  //Base classes
27:  //base classes are the tabs that will be loaded by the derived class
28:  //this makes it easy to add new tabs while keeping the amount that has
29:  //to go in one class to a minimum, especially since lvgl is not light
30:
31:
32:
33:  /**
34:   * @see: ../Styels.hpp
35:   * @see: ../../controller/controller.hpp
36:   *
37:   * contains general information about controllers and also allow
38:   * to test communication by sending rumbles and test strings
39:   */
40:  class GeneralControllerDebug : virtual Styles
41:  {
42:      private:
43:          lv_obj_t *container;
44:
45:          lv_obj_t *controller_column;
46:          lv_obj_t *connected_column;
47:          lv_obj_t *capacity_column;
48:          lv_obj_t *level_column;
49:
50:
51:
52:          lv_obj_t *btn_test_string;
53:          lv_obj_t *btn_test_string_label;
54:
55:          /**
56:           * @param: lv_obj_t* btn -> button that called the function
57:           * @return: lv_res_t -> return LV_RES_OK because object still exists
58:           * TODO: sometimes string does not send to the write location, fix
59:           *
60:           * sends a simple string to the controller lcd to test where output is
61:           * when the user clicks a button
62:           */
63:          static lv_res_t btn_test_string_action(lv_obj_t *btn);
64:
65:
66:
67:          lv_obj_t *btn_clear_scr;
68:          lv_obj_t *btn_clear_scr_label;
69:
70:          /**
71:           * @param: lv_obj_t* btn -> button that called the function
72:           * @return: lv_res_t -> return LV_RES_OK because object still exists
73:           * TODO: sometimes doesn't actually work
74:           *
75:           * clears any output on the lcd controller when user clicks a button
76:           */
77:          static lv_res_t btn_clear_scr_action(lv_obj_t *btn);
78:
79:
80:
81:
82:          lv_obj_t *btn_test_rumble;
83:          lv_obj_t *btn_test_rumble_label;
84:
85:          /**
86:           * @param: lv_obj_t* btn -> button that called the function
87:           * @return: lv_res_t -> return LV_RES_OK because object still exists
88:           *
89:           * tells controller to rumble
90:           */
91:          static lv_res_t btn_test_rumble_action(lv_obj_t *btn);
92:
93:
94:
95:          lv_obj_t *controls_info;
96:          lv_obj_t *motor2_info;
97:
98:      protected:
99:          /**
100:          * @return: None
101:          *
102:          * @see: ../../controller/controller.hpp
103:          *
104:          * updates data for the controllers such as connected or not, and battery
105:          */
106:         void update_general_info();
107:
108:     public:
109:         GeneralControllerDebug();
110:         virtual ~GeneralControllerDebug();
111:
112:         /**
113:          * @param: lv_obj_t* parent -> new parent for the main container
```

```
114:         * @return: None
115:         * TODO: depracate and fix method of inheritence, current method is not implemented well
116:         *
117:         * changes parent of containter
118:         */
119:        void GeneralControllerDebugInit(lv_obj_t *parent);
120:
121:    };
122:
123:
124:
125:
126:    /**
127:     * @see: ../Styels.hpp
128:     * @see: ../../controller/controller.hpp
129:     *
130:     * generic class for showing the functions or values of the controller for each button
131:     */
132:    class ControllerTab : virtual Styles
133:    {
134:        private:
135:            lv_obj_t *container;
136:
137:            //separated into two columns because LCD is not big enough
138:            //column one widgets
139:            lv_obj_t *button_names_one;
140:            lv_obj_t *button_col_one;
141:
142:            //column two widgets
143:            lv_obj_t *button_names_two;
144:            lv_obj_t *button_col_two;
145:
146:        public:
147:            ControllerTab(lv_obj_t *parent);
148:            virtual ~ControllerTab();
149:
150:
151:            /**
152:             * @param: pros::controller_id_e_t controller -> controller that the tab is looking at
153:             * @param: bool showing_values -> bool for if values should be shown or not
154:             * @return: None
155:             * TODO: make controller a member so that controller does not have to be a parameter
156:             *
157:             * shows either the values or the function the controller calls on the gui
158:             */
159:            void update(pros::controller_id_e_t controller, bool showing_values);
160:    };
161:
162:
163:
164:
165:
166:    //derived class
167:
168:
169:    /**
170:     * @see: ../Styles.hpp
171:     *
172:     * tab for showing data about controllers
173:     */
174:    class ControllerDebug :
175:        virtual private Styles,
176:        private GeneralControllerDebug
177:    {
178:        private:
179:            static bool showing_values;
180:
181:            //screen
182:            lv_obj_t *controller_debug_screen;
183:
184:            //title label
185:            lv_obj_t *title_label;
186:
187:
188:
189:            lv_obj_t *btn_back;
190:            lv_obj_t *btn_back_label;
191:
192:            /**
193:             * @param: lv_obj_t* btn -> button that called the funtion
194:             * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
195:             *
196:             * button callback function used to set cont to false meaning the
197:             * user wants to go to the title screen
198:             */
199:            static lv_res_t btn_back_action(lv_obj_t *btn);
200:
201:
202:
203:            lv_obj_t *btn_show_values;
204:            static lv_obj_t *btn_show_values_label;
205:
206:            /**
207:             * @param: lv_obj_t* btn -> button that called the funtion
208:             * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
209:             *
210:             * button callback function switch between showing the functions or values of
211:             * for the controller
212:             */
213:            static lv_res_t btn_show_values_action(lv_obj_t *btn);
214:
215:
216:
217:            static lv_obj_t *tabview; //tabview object
218:
219:            //individual tabs
220:            //content will come from base classes
221:            lv_obj_t *general_tab;
222:            lv_obj_t *master_tab;
223:            lv_obj_t *partner_tab;
224:
225:
226:        public:
```

```
227:        ControllerDebug();
228:        ~ControllerDebug();
229:
230:        static bool cont; //checks whether to keep letting user
231:                          //cycle through tabs
232:
233:
234:        /**
235:        * @return: None
236:        *
237:        * allows user to see information about the controller
238:        */
239:        void debug();
240:  };
241:
242:
243:
244:
245:  #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/ControllerDebug.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/16/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: ControllerDebug.hpp
8:   *
9:   * contains implementation for classes that show information about the controller
10:  */
11:
12:  #include "../../../../include/main.h"
13:  #include "../../../../include/api.h"
14:
15:  #include <unordered_map>
16:
17:  #include "../Styles.hpp"
18:  #include "../../controller/controller.hpp"
19:  #include "ControllerDebug.hpp"
20:
21:
22:  //declare static members of all classes
23:  bool ControllerDebug::showing_values = 0;
24:  bool ControllerDebug::cont = true;
25:  lv_obj_t *ControllerDebug::tabview;
26:  lv_obj_t *ControllerDebug::btn_show_values_label;
27:
28:
29:
30:  GeneralControllerDebug::GeneralControllerDebug()
31:  {
32:  //init container
33:      container = lv_cont_create(lv_scr_act(), NULL);
34:      lv_cont_set_fit(container, false, false);
35:      lv_obj_set_style(container, &gray);
36:      lv_cont_set_fit(container, false, false);
37:      lv_obj_set_width(container, CONTROLLER_CONTAINER_WIDTH);
38:      lv_obj_set_height(container, CONTROLLER_CONTAINER_HEIGHT);
39:
40:  //default text for each column
41:      std::string text1 = (
42:          "Controller\n"
43:          "Master\n"
44:          "Partner"
45:      );
46:
47:      std::string text2 = (
48:          "Connected\n"
49:          "no\n"
50:          "no"
51:      );
52:
53:      std::string text3 = (
54:          "Battery Capacity\n"
55:          "0\n"
56:          "0"
57:      );
58:
59:      std::string text4 = (
60:          "Battery Percent\n"
61:          "0\n"
62:          "0"
63:      );
64:
65:  //init controller column label
66:      controller_column = lv_label_create(container, NULL);
67:      lv_obj_set_style(controller_column, &toggle_tabbtn_pressed);
68:      lv_obj_set_width(controller_column, (CONTROLLER_CONTAINER_WIDTH / 4));
69:      lv_obj_set_height(controller_column, CONTROLLER_CONTAINER_HEIGHT);
70:      lv_label_set_align(controller_column, LV_LABEL_ALIGN_LEFT);
71:      lv_label_set_text(controller_column, text1.c_str());
72:
73:  //init connected column label
74:      connected_column = lv_label_create(container, NULL);
75:      lv_obj_set_style(connected_column, &toggle_tabbtn_pressed);
76:      lv_obj_set_width(connected_column, (CONTROLLER_CONTAINER_WIDTH / 4));
77:      lv_obj_set_height(connected_column, CONTROLLER_CONTAINER_HEIGHT);
78:      lv_label_set_align(connected_column, LV_LABEL_ALIGN_LEFT);
79:      lv_label_set_text(connected_column, text2.c_str());
80:
81:  //init capacity column label
82:      capacity_column = lv_label_create(container, NULL);
83:      lv_obj_set_style(capacity_column, &toggle_tabbtn_pressed);
84:      lv_obj_set_width(capacity_column, (CONTROLLER_CONTAINER_WIDTH / 4));
85:      lv_obj_set_height(capacity_column, CONTROLLER_CONTAINER_HEIGHT);
86:      lv_label_set_align(capacity_column, LV_LABEL_ALIGN_LEFT);
87:      lv_label_set_text(capacity_column, text3.c_str());
88:
89:  //init battery percentage column label
90:      level_column = lv_label_create(container, NULL);
91:      lv_obj_set_style(level_column, &toggle_tabbtn_pressed);
92:      lv_obj_set_width(level_column, (CONTROLLER_CONTAINER_WIDTH / 4));
93:      lv_obj_set_height(level_column, CONTROLLER_CONTAINER_HEIGHT);
94:      lv_label_set_align(level_column, LV_LABEL_ALIGN_LEFT);
95:      lv_label_set_text(level_column, text4.c_str());
96:
97:
98:  //init send test string button
99:      //button
100:     btn_test_string = lv_btn_create(container, NULL);
101:     lv_btn_set_style(btn_test_string, LV_BTN_STYLE_REL, &toggle_btn_released);
102:     lv_btn_set_style(btn_test_string, LV_BTN_STYLE_PR, &toggle_btn_pressed);
103:     lv_btn_set_action(btn_test_string, LV_BTN_ACTION_CLICK, btn_test_string_action);
104:     lv_obj_set_width(btn_test_string, 130);
105:     lv_obj_set_height(btn_test_string, 25);
106:
107:     //label
108:     btn_test_string_label = lv_label_create(btn_test_string, NULL);
109:     lv_obj_set_style(btn_test_string_label, &subheading_text);
110:     lv_label_set_text(btn_test_string_label, "Send Test String");
111:
112:
113: //init clear screen button
```

```
114:     //button
115:     btn_clear_scr = lv_btn_create(container, NULL);
116:     lv_btn_set_style(btn_clear_scr, LV_BTN_STYLE_REL, &toggle_btn_released);
117:     lv_btn_set_style(btn_clear_scr, LV_BTN_STYLE_PR, &toggle_btn_pressed);
118:     lv_btn_set_action(btn_clear_scr, LV_BTN_ACTION_CLICK, btn_clear_scr_action);
119:     lv_obj_set_width(btn_clear_scr, 130);
120:     lv_obj_set_height(btn_clear_scr, 25);
121:
122:     //label
123:     btn_clear_scr_label = lv_label_create(btn_clear_scr, NULL);
124:     lv_obj_set_style(btn_clear_scr_label, &subheading_text);
125:     lv_label_set_text(btn_clear_scr_label, "Clear Screen");
126:
127:
128:   //init send test rumble button
129:     //button
130:     btn_test_rumble = lv_btn_create(container, NULL);
131:     lv_btn_set_style(btn_test_rumble, LV_BTN_STYLE_REL, &toggle_btn_released);
132:     lv_btn_set_style(btn_test_rumble, LV_BTN_STYLE_PR, &toggle_btn_pressed);
133:     lv_btn_set_action(btn_test_rumble, LV_BTN_ACTION_CLICK, btn_test_rumble_action);
134:     lv_obj_set_width(btn_test_rumble, 130);
135:     lv_obj_set_height(btn_test_rumble, 25);
136:
137:     //label
138:     btn_test_rumble_label = lv_label_create(btn_test_rumble, NULL);
139:     lv_obj_set_style(btn_test_rumble_label, &subheading_text);
140:     lv_label_set_text(btn_test_rumble_label, "Send Test Rumble");
141:
142:
143:   //set positions relative to container
144:     lv_obj_align(controller_column, container, LV_ALIGN_IN_TOP_LEFT, 10, 10);
145:     lv_obj_align(connected_column, container, LV_ALIGN_IN_TOP_MID, -80, 10);
146:     lv_obj_align(capacity_column, container, LV_ALIGN_IN_TOP_MID, 15, 10);
147:     lv_obj_align(level_column, container, LV_ALIGN_IN_TOP_RIGHT, -30, 10);
148:
149:
150:     lv_obj_align(btn_test_string, container, LV_ALIGN_IN_BOTTOM_LEFT, 20, -30);
151:     lv_obj_align(btn_clear_scr, container, LV_ALIGN_IN_BOTTOM_MID, 0, -30);
152:     lv_obj_align(btn_test_rumble, container, LV_ALIGN_IN_BOTTOM_RIGHT, -20, -30);
153:
154:   }
155:
156:
157:   GeneralControllerDebug::~GeneralControllerDebug()
158:   {
159:
160:   }
161:
162:
163:
164:   /**
165:    * sends test string to controller
166:    */
167:   lv_res_t GeneralControllerDebug::btn_test_string_action(lv_obj_t *btn)
168:   {
169:     Controller::master.print(0, 0, "This is a test message");
170:     Controller::partner.print(0, 0, "This is a test message");
171:     return LV_RES_OK;
172:   }
173:
174:
175:   /**
176:    * clears the screen on the controller
177:    */
178:   lv_res_t GeneralControllerDebug::btn_clear_scr_action(lv_obj_t *btn)
179:   {
180:     Controller::master.print(0, 0, "");
181:     Controller::partner.print(0, 0, "");
182:     Controller::master.clear_line(0);
183:     Controller::partner.clear_line(0);
184:     return LV_RES_OK;
185:   }
186:
187:
188:   /**
189:    * sends a test rumble to the controller
190:    */
191:   lv_res_t GeneralControllerDebug::btn_test_rumble_action(lv_obj_t *btn)
192:   {
193:     Controller::master.rumble(". - - -");
194:     Controller::partner.rumble(". - - -");
195:     return LV_RES_OK;
196:   }
197:
198:
199:
200:
201:   /**
202:    * updates data on the tab
203:    */
204:   void GeneralControllerDebug::update_general_info()
205:   {
206:     std::string text1 = (
207:         "Controller\n"
208:         "Master\n"
209:         "Partner"
210:     );
211:
212:     std::string master_text = "no";
213:     std::string partner_text = "no";
214:     if ( Controller::master.is_connected() )
215:     {
216:         master_text = "yes";
217:     }
218:     if ( Controller::partner.is_connected() )
219:     {
220:         partner_text = "yes";
221:     }
222:
223:     std::string text2 = (
224:         "Connected\n"
225:         + master_text + "\n"
226:         + partner_text
```

```
227:     );
228:
229:
230:     std::string text3 = (
231:         "Battery Capacity\n"
232:         + std::to_string(Controller::master.get_battery_level()) + "\n"
233:         + std::to_string(Controller::partner.get_battery_level())
234:     );
235:
236:     std::string text4 = (
237:         "Battery Percent\n"
238:         + std::to_string(Controller::master.get_battery_capacity()) + "\n"
239:         + std::to_string(Controller::partner.get_battery_capacity())
240:     );
241:
242:     lv_label_set_text(controller_column, text1.c_str());
243:     lv_label_set_text(connected_column, text2.c_str());
244:     lv_label_set_text(capacity_column, text3.c_str());
245:     lv_label_set_text(level_column, text4.c_str());
246: }
247:
248:
249:
250: /**
251:  * changes parent of all objects
252:  */
253: void GeneralControllerDebug::GeneralControllerDebugInit(lv_obj_t *parent)
254: {
255:     //sets parent of container to pointer of new parent
256:     //this is to allow seperation of tabs into seperate classes
257:     //reduce the quantity in one class and to allow for ease of adding
258:     //new or different tabs
259:
260:     lv_obj_set_parent(container, parent);
261:
262: }
263:
264:
265:
266:
267: ControllerTab::ControllerTab(lv_obj_t *parent)
268: {
269: //init container
270:     container = lv_cont_create(parent, NULL);
271:     lv_cont_set_fit(container, false, false);
272:     lv_obj_set_style(container, &gray);
273:     lv_cont_set_fit(container, false, false);
274:     lv_obj_set_width(container, CONTROLLER_CONTAINER_WIDTH);
275:     lv_obj_set_height(container, CONTROLLER_CONTAINER_HEIGHT);
276:
277: //text for names
278:     std::string ctrl_col1 = (
279:         "Analog Left X\n"
280:         "Analog Left Y\n"
281:         "Analog Right X\n"
282:         "Analog Right Y\n"
283:         "Digital L1\n"
284:         "Digital L2\n"
285:         "Digital R1\n"
286:         "Digital R2"
287:     );
288:
289:     std::string ctrl_col2 = (
290:         "Digital Up\n"
291:         "Digital Down\n"
292:         "Digital Left\n"
293:         "Digital Right\n"
294:         "Digital X\n"
295:         "Digital B\n"
296:         "Digital Y\n"
297:         "Digital A\n"
298:     );
299:
300: //column one button names
301:     button_names_one = lv_label_create(container, NULL);
302:     lv_label_set_style(button_names_one, &toggle_tabbtn_pressed);
303:     lv_obj_set_width(button_names_one, CONTROLLER_CONTAINER_WIDTH / 4);
304:     lv_obj_set_height(button_names_one, 20);
305:     lv_label_set_align(button_names_one, LV_LABEL_ALIGN_LEFT);
306:     lv_label_set_text(button_names_one, ctrl_col1.c_str());
307:
308:
309: //column two button names
310:     button_names_two = lv_label_create(container, NULL);
311:     lv_label_set_style(button_names_two, &toggle_tabbtn_pressed);
312:     lv_obj_set_width(button_names_two, CONTROLLER_CONTAINER_WIDTH / 4);
313:     lv_obj_set_height(button_names_two, 20);
314:     lv_label_set_align(button_names_two, LV_LABEL_ALIGN_LEFT);
315:     lv_label_set_text(button_names_two, ctrl_col2.c_str());
316:
317:
318: //column one second part that contains function or values
319:     button_col_one = lv_label_create(container, NULL);
320:     lv_label_set_style(button_col_one, &toggle_tabbtn_pressed);
321:     lv_obj_set_width(button_col_one, CONTROLLER_CONTAINER_WIDTH / 4);
322:     lv_obj_set_height(button_col_one, 20);
323:     lv_label_set_align(button_col_one, LV_LABEL_ALIGN_LEFT);
324:
325:
326: //column two second part that contains function or values
327:     button_col_two = lv_label_create(container, NULL);
328:     lv_label_set_style(button_col_two, &toggle_tabbtn_pressed);
329:     lv_obj_set_width(button_col_two, CONTROLLER_CONTAINER_WIDTH / 4);
330:     lv_obj_set_height(button_col_two, 20);
331:     lv_label_set_align(button_col_two, LV_LABEL_ALIGN_LEFT);
332:
333:
334: //set positions relative to container
335:     lv_obj_align(button_names_one, container, LV_ALIGN_IN_TOP_LEFT, 10, 10);
336:     lv_obj_align(button_col_one, container, LV_ALIGN_IN_TOP_MID, -80, 10);
337:
338:     lv_obj_align(button_names_two, container, LV_ALIGN_IN_TOP_MID, 60, 10);
339:     lv_obj_align(button_col_two, container, LV_ALIGN_IN_TOP_RIGHT, -70, 10);
```

```
340:
341: }
342:
343:
344: ControllerTab::~ControllerTab()
345: {
346:     lv_obj_del(button_names_one);
347:     lv_obj_del(button_names_two);
348:     lv_obj_del(button_col_one);
349:     lv_obj_del(button_col_two);
350:
351:     lv_obj_del(container);
352: }
353:
354:
355:
356: /**
357:  * updates the data for the controller tab for either the value of each button
358:  * or the function each button performs by looking at an std::unordered_map contained in
359:  * controller.hpp
360:  */
361: void ControllerTab::update(pros::controller_id_e_t controller, bool showing_values)
362: {
363:     std::string functions_col1 = "";
364:     std::string functions_col2 = "";
365:
366:     std::string values_col1 = "";
367:     std::string values_col2 = "";
368:
369:     if ( controller == pros::E_CONTROLLER_MASTER )
370:     {
371:         //text for functions
372:         functions_col1 = (
373:             Controller::MASTER_CONTROLLER_ANALOG_MAPPINGS.at(pros::E_CONTROLLER_ANALOG_LEFT_X) + "\n"
374:             + Controller::MASTER_CONTROLLER_ANALOG_MAPPINGS.at(pros::E_CONTROLLER_ANALOG_LEFT_Y) + "\n"
375:             + Controller::MASTER_CONTROLLER_ANALOG_MAPPINGS.at(pros::E_CONTROLLER_ANALOG_RIGHT_X) + "\n"
376:             + Controller::MASTER_CONTROLLER_ANALOG_MAPPINGS.at(pros::E_CONTROLLER_ANALOG_RIGHT_Y) + "\n"
377:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_L1) + "\n"
378:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_L2) + "\n"
379:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_R2) + "\n"
380:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_R1)
381:         );
382:
383:         functions_col2 = (
384:             Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_UP) + "\n"
385:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_DOWN) + "\n"
386:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_LEFT) + "\n"
387:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_RIGHT) + "\n"
388:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_X) + "\n"
389:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_B) + "\n"
390:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_Y) + "\n"
391:             + Controller::MASTER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_A)
392:         );
393:
394:         values_col1 = (
395:             std::to_string(Controller::master.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_X)) + "\n"
396:             + std::to_string(Controller::master.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y)) + "\n"
397:             + std::to_string(Controller::master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X)) + "\n"
398:             + std::to_string(Controller::master.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_Y)) + "\n"
399:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_L1)) + "\n"
400:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_L2)) + "\n"
401:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_R2)) + "\n"
402:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_R1))
403:         );
404:
405:         values_col2 = (
406:             std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_UP)) + "\n"
407:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_DOWN)) + "\n"
408:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_LEFT)) + "\n"
409:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_RIGHT)) + "\n"
410:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_X)) + "\n"
411:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_B)) + "\n"
412:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_Y)) + "\n"
413:             + std::to_string(Controller::master.get_digital(pros::E_CONTROLLER_DIGITAL_A))
414:         );
415:
416:
417:     }
418:
419:     else if ( controller == pros::E_CONTROLLER_PARTNER )
420:     {
421:         //text for functions
422:         functions_col1 = (
423:             Controller::PARTNER_CONTROLLER_ANALOG_MAPPINGS.at(pros::E_CONTROLLER_ANALOG_LEFT_X) + "\n"
424:             + Controller::PARTNER_CONTROLLER_ANALOG_MAPPINGS.at(pros::E_CONTROLLER_ANALOG_LEFT_Y) + "\n"
425:             + Controller::PARTNER_CONTROLLER_ANALOG_MAPPINGS.at(pros::E_CONTROLLER_ANALOG_RIGHT_X) + "\n"
426:             + Controller::PARTNER_CONTROLLER_ANALOG_MAPPINGS.at(pros::E_CONTROLLER_ANALOG_RIGHT_Y) + "\n"
427:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_L1) + "\n"
428:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_L2) + "\n"
429:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_R2) + "\n"
430:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_R1)
431:         );
432:
433:         functions_col2 = (
434:             Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_UP) + "\n"
435:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_DOWN) + "\n"
436:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_LEFT) + "\n"
437:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_RIGHT) + "\n"
438:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_X) + "\n"
439:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_B) + "\n"
440:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_Y) + "\n"
441:             + Controller::PARTNER_CONTROLLER_DIGITAL_MAPPINGS.at(pros::E_CONTROLLER_DIGITAL_A)
442:         );
443:
444:
445:         values_col1 = (
446:             std::to_string(Controller::partner.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_X)) + "\n"
447:             + std::to_string(Controller::partner.get_analog(pros::E_CONTROLLER_ANALOG_LEFT_Y)) + "\n"
448:             + std::to_string(Controller::partner.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_X)) + "\n"
449:             + std::to_string(Controller::partner.get_analog(pros::E_CONTROLLER_ANALOG_RIGHT_Y)) + "\n"
450:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_L1)) + "\n"
451:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_L2)) + "\n"
452:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_R2)) + "\n"
```

```
453:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_R1))
454:         );
455:
456:         values_col2 = (
457:             std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_UP)) + "\n"
458:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_DOWN)) + "\n"
459:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_LEFT)) + "\n"
460:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_RIGHT)) + "\n"
461:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_X)) + "\n"
462:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_B)) + "\n"
463:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_Y)) + "\n"
464:             + std::to_string(Controller::partner.get_digital(pros::E_CONTROLLER_DIGITAL_A))
465:         );
466:     }
467:
468:     if ( showing_values )
469:     {
470:         lv_label_set_text(button_col_one, values_col1.c_str());
471:         lv_label_set_text(button_col_two, values_col2.c_str());
472:     }
473:
474:     else
475:     {
476:         lv_label_set_text(button_col_one, functions_col1.c_str());
477:         lv_label_set_text(button_col_two, functions_col2.c_str());
478:     }
479:
480: }
481:
482:
483:
484:
485:
486: ControllerDebug::ControllerDebug()
487: {
488: //reset statics
489:     cont = true;
490:     showing_values = 0;
491:
492:
493: //init screen
494:     controller_debug_screen = lv_obj_create(NULL, NULL);
495:     lv_obj_set_style(controller_debug_screen, &gray);
496:
497: //init title label
498:     title_label = lv_label_create(controller_debug_screen, NULL);
499:     lv_label_set_style(title_label, &heading_text);
500:     lv_obj_set_width(title_label, CONTROLLER_CONTAINER_WIDTH);
501:     lv_obj_set_height(title_label, 20);
502:     lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
503:     lv_label_set_text(title_label, "Controller - Debug");
504:
505: //init tabview
506:     tabview = lv_tabview_create(controller_debug_screen, NULL);
507:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BG, &gray);
508:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BTN_REL, &toggle_tabbtn_released);
509:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BTN_PR, &toggle_tabbtn_pressed);
510:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_INDIC, &sw_indic);
511:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BTN_TGL_REL, &toggle_tabbtn_pressed);
512:     //lv_tabview_set_tab_load_action(tabview,  tab_load_action);
513:     lv_obj_set_width(tabview, CONTROLLER_CONTAINER_WIDTH);
514:     lv_obj_set_height(tabview, 200);
515:
516: //init tabs
517:     general_tab = lv_tabview_add_tab(tabview, "General");
518:     master_tab = lv_tabview_add_tab(tabview, "Master Controller");
519:     partner_tab = lv_tabview_add_tab(tabview, "Partner Controller");
520:
521:
522: //init back button
523:     //button
524:     btn_back = lv_btn_create(controller_debug_screen, NULL);
525:     lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
526:     lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
527:     lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
528:     lv_obj_set_width(btn_back, 75);
529:     lv_obj_set_height(btn_back, 25);
530:
531:     //label
532:     btn_back_label = lv_label_create(btn_back, NULL);
533:     lv_obj_set_style(btn_back_label, &heading_text);
534:     lv_label_set_text(btn_back_label, "Back");
535:
536:
537: //init button to switch between showing values and functions
538:     //button
539:     btn_show_values = lv_btn_create(controller_debug_screen, NULL);
540:     lv_btn_set_style(btn_show_values, LV_BTN_STYLE_REL, &toggle_btn_released);
541:     lv_btn_set_style(btn_show_values, LV_BTN_STYLE_PR, &toggle_btn_pressed);
542:     lv_btn_set_action(btn_show_values, LV_BTN_ACTION_CLICK, btn_show_values_action);
543:     lv_obj_set_width(btn_show_values, 150);
544:     lv_obj_set_height(btn_show_values, 25);
545:
546:     //label
547:     btn_show_values_label = lv_label_create(btn_show_values, NULL);
548:     lv_obj_set_style(btn_show_values_label, &heading_text);
549:     lv_label_set_text(btn_show_values_label, "Show Values");
550:
551:     lv_obj_set_hidden(btn_show_values, true);  //set hidden because button is
552:                                 //not needed on the default tab
553:
554:
555: //init tabs from other classes
556:     GeneralControllerDebugInit(general_tab);
557:
558:
559: //set positions
560:     lv_obj_set_pos(btn_back, 30, 210);
561:     lv_obj_align(btn_show_values, controller_debug_screen, LV_ALIGN_IN_BOTTOM_MID, 0, -5);
562:
563:     lv_obj_set_pos(title_label, 180, 5);
564:
565:     lv_obj_set_pos(tabview, 20, 25);
```

```
566: }
567:
568:
569:
570:
571: ControllerDebug::~ControllerDebug()
572: {
573:     lv_obj_del(controller_debug_screen);
574: }
575:
576:
577:
578:
579: /**
580:  * switches between showing values or function by setting variable to the
581:  * opposite of itself and then it updates the text label based on the new value
582:  */
583:
584: lv_res_t ControllerDebug::btn_show_values_action(lv_obj_t *btn)
585: {
586:     showing_values = !showing_values;
587:     if ( showing_values )
588:     {
589:         lv_label_set_text(btn_show_values_label, "Show Functions");
590:     }
591:     else
592:     {
593:         lv_label_set_text(btn_show_values_label, "Show Values");
594:     }
595:
596:     return LV_RES_OK;
597: }
598:
599:
600:
601:
602: /**
603:  * callback funciton that exits main loop when button is pressed
604:  */
605: lv_res_t ControllerDebug::btn_back_action(lv_obj_t *btn)
606: {
607:     cont = false;
608:     return LV_RES_OK;
609: }
610:
611:
612:
613:
614: /**
615:  * main loop that updates controller information
616:  */
617: void ControllerDebug::debug()
618: {
619:     //used to check if user wants to continue cycling through
620:     //tabs. Will be set to zero and loop will break if user hits
621:     //the back button
622:     cont = true;
623:
624:     lv_tabview_set_tab_act(tabview, 0, NULL);
625:     lv_scr_load(controller_debug_screen);
626:
627:     //init tabs from other classes
628:     ControllerTab controller_tab(master_tab);
629:     ControllerTab controller_tab2(partner_tab);
630:
631:     while ( cont )
632:     {
633:         switch ( lv_tabview_get_tab_act(tabview) ) //switches to tab user wants to go to
634:         {
635:             case 0:
636:                 lv_obj_set_hidden(btn_show_values, true);
637:                 update_general_info();
638:                 break;
639:             case 1:
640:                 lv_obj_set_hidden(btn_show_values, false);
641:                 controller_tab.update(pros::E_CONTROLLER_MASTER, showing_values);
642:                 break;
643:             case 2:
644:                 lv_obj_set_hidden(btn_show_values, false);
645:                 controller_tab2.update(pros::E_CONTROLLER_PARTNER, showing_values);
646:                 break;
647:         }
648:         pros::delay(200);
649:     }
650:
651: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/FieldControlDebug.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/16/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * contains class with methods that allow the user to see info about the state
8:   * od the field control
9:   */
10:
11: #ifndef __FIELDCONTROLDEBUG_HPP__
12: #define __FIELDCONTROLDEBUG_HPP__
13:
14:
15: #include "../../../../include/main.h"
16:
17: #include "../Styles.hpp"
18:
19:
20: /**
21:  * @see: ../Styles.hpp
22:  *
23:  * contains methods that show user data about the field control
24:  */
25: class FieldControlDebug : private Styles
26: {
27:     private:
28:         lv_obj_t *field_ctrl_screen;
29:         lv_obj_t *title_label;
30:
31:         lv_obj_t *labels_label;
32:         lv_obj_t *info_label;
33:
34:         //back button
35:         lv_obj_t *btn_back;
36:         lv_obj_t *btn_back_label;
37:
38:         /**
39:          * @param: lv_obj_t* btn -> button that called the funtion
40:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
41:          *
42:          * button callback function used to set cont to false meaning the
43:          * user wants to go to the title screen
44:          */
45:         static lv_res_t btn_back_action(lv_obj_t *btn);
46:
47:     public:
48:         static bool cont;
49:
50:         FieldControlDebug();
51:         ~FieldControlDebug();
52:
53:
54:         /**
55:          * @return: None
56:          * TODO: use ternary operator to condense and make more readable
57:          *
58:          * allows user to see information about the state of field control
59:          */
60:         void debug();
61:
62: };
63:
64:
65:
66:
67: #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/FieldControlDebug.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/16/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: FieldControlDebug.hpp
8:   *
9:   * contains implementation for class with field control data
10:  */
11:
12:  #include "../../../../include/main.h"
13:  #include "../../../../include/api.h"
14:
15:  #include "../Styles.hpp"
16:  #include "FieldControlDebug.hpp"
17:
18:
19:  bool FieldControlDebug::cont = true;
20:
21:  FieldControlDebug::FieldControlDebug()
22:  {
23:      cont = true;
24:
25:  //screen
26:      field_ctrl_screen = lv_obj_create(NULL, NULL);
27:      lv_obj_set_style(field_ctrl_screen, &gray);
28:
29:  //init back button
30:      //button
31:      btn_back = lv_btn_create(field_ctrl_screen, NULL);
32:      lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
33:      lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
34:      lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
35:      lv_obj_set_width(btn_back, 75);
36:      lv_obj_set_height(btn_back, 25);
37:
38:      //label
39:      btn_back_label = lv_label_create(btn_back, NULL);
40:      lv_obj_set_style(btn_back_label, &heading_text);
41:      lv_label_set_text(btn_back_label, "Back");
42:
43:  //init title label
44:      title_label = lv_label_create(field_ctrl_screen, NULL);
45:      lv_label_set_style(title_label, &heading_text);
46:      lv_obj_set_width(title_label, 440);
47:      lv_obj_set_height(title_label, 20);
48:      lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
49:      lv_label_set_text(title_label, "Field Control - Debug");
50:
51:  //init headers label
52:      labels_label = lv_label_create(field_ctrl_screen, NULL);
53:      lv_label_set_style(labels_label, &subheading_text);
54:      lv_obj_set_width(labels_label, 220);
55:      lv_obj_set_height(labels_label, 200);
56:      lv_label_set_align(labels_label, LV_LABEL_ALIGN_LEFT);
57:
58:      std::string labels_label_text = (
59:              "connected to competition switch\n"
60:              "disabled\n"
61:              "game state"
62:      );
63:
64:      lv_label_set_text(labels_label, labels_label_text.c_str());
65:
66:  //init values label
67:      info_label = lv_label_create(field_ctrl_screen, NULL);
68:      lv_label_set_style(info_label, &subheading_text);
69:      lv_obj_set_width(info_label, 220);
70:      lv_obj_set_height(info_label, 200);
71:      lv_label_set_align(info_label, LV_LABEL_ALIGN_LEFT);
72:
73:      std::string info_label_text = (
74:          "no\n"
75:          "no\n"
76:          "no"
77:      );
78:
79:      lv_label_set_text(info_label, info_label_text.c_str());
80:
81:  //set positions
82:      lv_obj_set_pos(btn_back, 30, 210);
83:
84:      lv_obj_align(title_label, field_ctrl_screen, LV_ALIGN_IN_TOP_MID, 0, 10);
85:
86:      lv_obj_set_pos(labels_label, 20, 40);
87:      lv_obj_set_pos(info_label, 360, 40);
88:
89:
90:  }
91:
92:  FieldControlDebug::~FieldControlDebug()
93:  {
94:      lv_obj_del(field_ctrl_screen);
95:  }
96:
97:
98:  /**
99:   * sets cont to false to break main loop so main function returns
100:  */
101: lv_res_t FieldControlDebug::btn_back_action(lv_obj_t *btn)
102: {
103:     cont = false;
104:     return LV_RES_OK;
105: }
106:
107:
108: /**
109:  * main loop that updates the field control information
110:  */
111: void FieldControlDebug::debug()
112: {
113:     cont = true;
```

```
114:
115:    lv_scr_load(field_ctrl_screen);
116:
117:    while ( cont )
118:    {
119:       std::string info_label_text = "";
120:
121:       if ( pros::competition::is_connected() )
122:       {
123:          info_label_text = info_label_text + "yes\n";
124:       }
125:       else
126:       {
127:          info_label_text = info_label_text + "no\n";
128:       }
129:
130:       if ( pros::competition::is_disabled() )
131:       {
132:          info_label_text = info_label_text + "yes\n";
133:       }
134:
135:       else
136:       {
137:          info_label_text = info_label_text + "no\n";
138:       }
139:
140:       if ( pros::competition::is_autonomous() )
141:       {
142:          info_label_text = info_label_text + "autonomous\n";
143:       }
144:
145:       else
146:       {
147:          info_label_text = info_label_text + "driver control\n";
148:       }
149:
150:       lv_label_set_text(info_label, info_label_text.c_str());
151:
152:
153:       pros::delay(100);
154:    }
155: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/InternalMotorDebug.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on:
5:   * @reviewed_by:
6:   *
7:   * contains class that for debugging the internal Motor PID constants
8:   * from the lcd without having to recompile and upload code
9:   */
10:
11: #ifndef __INTERNALMOTORDEBUG_HPP__
12: #define __INTERNALMOTORDEBUG_HPP__
13:
14: #include "main.h"
15:
16: #include "../Styles.hpp"
17: #include "../../../Configuration.hpp"
18: #include "../../motors/Motor.hpp"
19:
20:
21: /**
22:  * @see: ../../motors/Motor.hpp
23:  *
24:  * allows user to tune pid constants by logging data and running unit tests
25:  * with given parameters
26:  */
27: class InternalMotorDebug : private Styles
28: {
29:     private:
30:         static bool cont;
31:         static bool run;
32:
33:         Motor motor;
34:
35:         lv_obj_t *main_screen;
36:         lv_obj_t *title_label;
37:
38:     //parameter labels side one
39:         lv_obj_t *kp_label;
40:         lv_obj_t *kp_text_area;
41:
42:         lv_obj_t *ki_label;
43:         lv_obj_t *ki_text_area;
44:
45:         lv_obj_t *kd_label;
46:         lv_obj_t *kd_text_area;
47:
48:         lv_obj_t *I_max_label;
49:         lv_obj_t *I_max_text_area;
50:
51:         lv_obj_t *slew_label;
52:         lv_obj_t *slew_text_area;
53:
54:         lv_obj_t *setpoint_label;
55:         lv_obj_t *setpoint_text_area;
56:
57:         lv_obj_t *duration_label;
58:         lv_obj_t *duration_text_area;
59:
60:     //parameter labels side two
61:         //port
62:         lv_obj_t *port_label;
63:         lv_obj_t *port_text_area;
64:
65:         //gearset
66:         static pros::motor_gearset_e_t current_gearset;
67:         lv_obj_t *gearset_label;
68:         lv_obj_t *ddlist_gearset;
69:
70:         /**
71:          * @param: lv_obj_t* ddlist -> the dropdown list object for the callback function
72:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
73:          *
74:          * sets the brake mode for the motor set which will be updated in the main loop
75:          */
76:         static lv_res_t ddlist_gearset_action(lv_obj_t *ddlist);
77:
78:         //brake mode
79:         static pros::motor_brake_mode_e_t current_brake_mode;
80:         lv_obj_t *brakemode_label;
81:         lv_obj_t *ddlist_brake_mode;
82:
83:         /**
84:          * @param: lv_obj_t* ddlist -> the dropdown list object for the callback function
85:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
86:          *
87:          * sets the brake mode for the motor set which will be updated in the main loop
88:          */
89:         static lv_res_t ddlist_brake_mode_action(lv_obj_t *ddlist);
90:
91:     //information label
92:         lv_obj_t *information_label;
93:
94:     //keyboard
95: /*      lv_obj_t *keyboard;*/
96:
97:
98:     //back button
99:         lv_obj_t *btn_back;
100:        lv_obj_t *btn_back_label;
101:
102:        /**
103:         * @param: lv_obj_t* btn -> button that called the funtion
104:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
105:         *
106:         * button callback function used to set cont to false meaning the
107:         * user wants to go to the title screen
108:         */
109:        static lv_res_t btn_back_action(lv_obj_t *btn);
110:
111:
112:    //run unit test button
113:        lv_obj_t *btn_run;
```

```
114:        lv_obj_t *btn_run_label;
115:
116:        /**
117:         * @param: lv_obj_t* btn -> button that called the funtion
118:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
119:         *
120:         * button to run unit_test and log data with the given parameters
121:         */
122:        static lv_res_t btn_run_action(lv_obj_t *btn);
123:
124:    //actual unit test function
125:        int run_unit_test( );
126:
127:
128:    public:
129:        InternalMotorDebug();
130:        ~InternalMotorDebug();
131:
132:        /**
133:         * @return: None
134:         *
135:         * -
136:         */
137:        void debug();
138:
139:  };
140:
141:
142:
143:  #endif
```

```
/**
 * @file: ./RobotCode/src/lcdCode/Debug/InternalMotorDebug.cpp
 * @author: Aiden Carney
 * @reviewed_on:
 * @reviewed_by:
 *
 * @see: InternalMotorDebug.hpp
 *
 * contains class implementation for tuning the motors internal velocity PID
 * controller
 */

#include <stdexcept>
#include <string>

#include "main.h"

#include "InternalMotorDebug.hpp"
#include "../Styles.hpp"
#include "../../../Configuration.hpp"
#include "../../motors/Motor.hpp"
#include "../../motors/MotorThread.hpp"


bool InternalMotorDebug::cont = true;
bool InternalMotorDebug::run = false;
pros::motor_gearset_e_t InternalMotorDebug::current_gearset = pros::E_MOTOR_GEARSET_18;
pros::motor_brake_mode_e_t InternalMotorDebug::current_brake_mode = pros::E_MOTOR_BRAKE_COAST;


InternalMotorDebug::InternalMotorDebug() :
    motor(1, pros::E_MOTOR_GEARSET_18, false)
{
    MotorThread* motor_thread = MotorThread::get_instance();
    motor_thread->register_motor(motor);


    cont = true;
    run = false;

//screen
    main_screen = lv_obj_create(NULL, NULL);
    lv_obj_set_style(main_screen, &gray);

//init title label
    title_label = lv_label_create(main_screen, NULL);
    lv_label_set_style(title_label, &heading_text);
    lv_obj_set_width(title_label, 440);
    lv_obj_set_height(title_label, 20);
    lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
    lv_label_set_text(title_label, "Velocity PID Controller - Debugger");


    //init parameters side two
    //port
        port_label = lv_label_create(main_screen, NULL);
        lv_label_set_style(port_label, &heading_text);
        lv_obj_set_width(port_label, 100);
        lv_obj_set_height(port_label, 40);
        lv_label_set_align(port_label, LV_LABEL_ALIGN_LEFT);
        lv_label_set_text(port_label, "Port");

        port_text_area = lv_ta_create(main_screen, NULL);
        lv_obj_set_style(port_text_area, &subheading_text);
        lv_ta_set_accepted_chars(port_text_area, "0123456789");
        lv_obj_set_size(port_text_area, 80, 20);
        lv_ta_set_text(port_text_area, "1");
        lv_ta_set_one_line(port_text_area, true);

    //gearset
        current_gearset = pros::E_MOTOR_GEARSET_18;

        gearset_label = lv_label_create(main_screen, NULL);
        lv_label_set_style(gearset_label, &heading_text);
        lv_obj_set_width(gearset_label, 100);
        lv_obj_set_height(gearset_label, 40);
        lv_label_set_align(gearset_label, LV_LABEL_ALIGN_LEFT);
        lv_label_set_text(gearset_label, "Gearset");

        ddlist_gearset = lv_ddlist_create(main_screen, NULL);
        lv_ddlist_set_options(ddlist_gearset, "100\n200\n600");
        lv_obj_set_style(ddlist_gearset, &subheading_text);
        lv_obj_set_width(ddlist_gearset, 125);
        lv_obj_set_height(ddlist_gearset, 60);
        lv_ddlist_set_action(ddlist_gearset, ddlist_gearset_action);
        lv_ddlist_set_selected(ddlist_gearset, 1);

    //brakemode
        current_brake_mode = pros::E_MOTOR_BRAKE_COAST;

        brakemode_label = lv_label_create(main_screen, NULL);
        lv_label_set_style(brakemode_label, &heading_text);
        lv_obj_set_width(brakemode_label, 100);
        lv_obj_set_height(brakemode_label, 40);
        lv_label_set_align(brakemode_label, LV_LABEL_ALIGN_LEFT);
        lv_label_set_text(brakemode_label, "Brakemode");

        ddlist_brake_mode = lv_ddlist_create(main_screen, NULL);
        lv_ddlist_set_options(ddlist_brake_mode, "Coast\n"
                                    "Brake\n"
                                    "PID Hold");
        lv_obj_set_style(ddlist_brake_mode, &subheading_text);
        lv_obj_set_width(ddlist_brake_mode, 125);
        lv_obj_set_height(ddlist_brake_mode, 20);
        lv_ddlist_set_action(ddlist_brake_mode, ddlist_brake_mode_action);


//init parameters side one
//kP
    kp_label = lv_label_create(main_screen, NULL);
    lv_label_set_style(kp_label, &heading_text);
    lv_obj_set_width(kp_label, 440);
```

```
114:     lv_obj_set_height(kp_label, 20);
115:     lv_label_set_align(kp_label, LV_LABEL_ALIGN_LEFT);
116:     lv_label_set_text(kp_label, "kP");
117:
118:     kp_text_area = lv_ta_create(main_screen, NULL);
119:     lv_obj_set_style(kp_text_area, &subheading_text);
120:     lv_ta_set_accepted_chars(kp_text_area, ".0123456789");
121:     lv_obj_set_size(kp_text_area, 80, 15);
122:     lv_ta_set_text(kp_text_area, std::to_string(motor.get_pid().kP).c_str());
123:     lv_ta_set_one_line(kp_text_area, true);
124:
125:   //kI
126:     ki_label = lv_label_create(main_screen, NULL);
127:     lv_label_set_style(ki_label, &heading_text);
128:     lv_obj_set_width(ki_label, 100);
129:     lv_obj_set_height(ki_label, 20);
130:     lv_label_set_align(ki_label, LV_LABEL_ALIGN_LEFT);
131:     lv_label_set_text(ki_label, "kI");
132:
133:     ki_text_area = lv_ta_create(main_screen, NULL);
134:     lv_obj_set_style(ki_text_area, &subheading_text);
135:     lv_ta_set_accepted_chars(ki_text_area, ".0123456789");
136:     lv_obj_set_size(ki_text_area, 80, 15);
137:     lv_ta_set_text(ki_text_area, std::to_string(motor.get_pid().kI).c_str());
138:     lv_ta_set_one_line(ki_text_area, true);
139:
140:   //kD
141:     kd_label = lv_label_create(main_screen, NULL);
142:     lv_label_set_style(kd_label, &heading_text);
143:     lv_obj_set_width(kd_label, 100);
144:     lv_obj_set_height(kd_label, 20);
145:     lv_label_set_align(kd_label, LV_LABEL_ALIGN_LEFT);
146:     lv_label_set_text(kd_label, "kD");
147:
148:     kd_text_area = lv_ta_create(main_screen, NULL);
149:     lv_obj_set_style(kd_text_area, &subheading_text);
150:     lv_ta_set_accepted_chars(kd_text_area, ".0123456789");
151:     lv_obj_set_size(kd_text_area, 80, 15);
152:     lv_ta_set_text(kd_text_area, std::to_string(motor.get_pid().kD).c_str());
153:     lv_ta_set_one_line(kd_text_area, true);
154:
155:   //I max
156:     I_max_label = lv_label_create(main_screen, NULL);
157:     lv_label_set_style(I_max_label, &heading_text);
158:     lv_obj_set_width(I_max_label, 100);
159:     lv_obj_set_height(I_max_label, 20);
160:     lv_label_set_align(I_max_label, LV_LABEL_ALIGN_LEFT);
161:     lv_label_set_text(I_max_label, "kI Max");
162:
163:     I_max_text_area = lv_ta_create(main_screen, NULL);
164:     lv_obj_set_style(I_max_text_area, &subheading_text);
165:     lv_ta_set_accepted_chars(I_max_text_area, ".0123456789");
166:     lv_obj_set_size(I_max_text_area, 80, 15);
167:     lv_ta_set_text(I_max_text_area, std::to_string(motor.get_pid().I_max).c_str());
168:     lv_ta_set_one_line(I_max_text_area, true);
169:
170:   //slew rate
171:     slew_label = lv_label_create(main_screen, NULL);
172:     lv_label_set_style(slew_label, &heading_text);
173:     lv_obj_set_width(slew_label, 100);
174:     lv_obj_set_height(slew_label, 40);
175:     lv_label_set_align(slew_label, LV_LABEL_ALIGN_LEFT);
176:     lv_label_set_text(slew_label, "Slew Rate");
177:
178:     slew_text_area = lv_ta_create(main_screen, NULL);
179:     lv_obj_set_style(slew_text_area, &subheading_text);
180:     lv_ta_set_accepted_chars(slew_text_area, "0123456789");
181:     lv_obj_set_size(slew_text_area, 80, 15);
182:     lv_ta_set_text(slew_text_area, std::to_string(motor.get_slew_rate()).c_str());
183:     lv_ta_set_one_line(slew_text_area, true);
184:
185:   //setpoint_label
186:     setpoint_label = lv_label_create(main_screen, NULL);
187:     lv_label_set_style(setpoint_label, &heading_text);
188:     lv_obj_set_width(setpoint_label, 100);
189:     lv_obj_set_height(setpoint_label, 40);
190:     lv_label_set_align(setpoint_label, LV_LABEL_ALIGN_LEFT);
191:     lv_label_set_text(setpoint_label, "Setpoint");
192:
193:     setpoint_text_area = lv_ta_create(main_screen, NULL);
194:     lv_obj_set_style(setpoint_text_area, &subheading_text);
195:     lv_ta_set_accepted_chars(setpoint_text_area, "0123456789");
196:     lv_obj_set_size(setpoint_text_area, 80, 15);
197:     lv_ta_set_text(setpoint_text_area, "200");
198:     lv_ta_set_one_line(setpoint_text_area, true);
199:
200:   //duration
201:     duration_label = lv_label_create(main_screen, NULL);
202:     lv_label_set_style(duration_label, &heading_text);
203:     lv_obj_set_width(duration_label, 100);
204:     lv_obj_set_height(duration_label, 40);
205:     lv_label_set_align(duration_label, LV_LABEL_ALIGN_LEFT);
206:     lv_label_set_text(duration_label, "Duration");
207:
208:     duration_text_area = lv_ta_create(main_screen, NULL);
209:     lv_obj_set_style(duration_text_area, &subheading_text);
210:     lv_ta_set_accepted_chars(duration_text_area, "0123456789");
211:     lv_obj_set_size(duration_text_area, 80, 15);
212:     lv_ta_set_text(duration_text_area, "10000");
213:     lv_ta_set_one_line(duration_text_area, true);
214:
215:
216:
217:   //information label
218:     information_label = lv_label_create(main_screen, NULL);
219:     lv_obj_set_style(information_label, &subheading_text);
220:     lv_label_set_text(information_label, "Info");
221:
222:
223:   //init back button
224:     //button
225:     btn_back = lv_btn_create(main_screen, NULL);
226:     lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
```

```
227:     lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
228:     lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
229:     lv_obj_set_width(btn_back, 75);
230:     lv_obj_set_height(btn_back, 25);
231:
232:     //label
233:     btn_back_label = lv_label_create(btn_back, NULL);
234:     lv_obj_set_style(btn_back_label, &heading_text);
235:     lv_label_set_text(btn_back_label, "Back");
236:
237:
238:   //init run button
239:     //button
240:     btn_run = lv_btn_create(main_screen, NULL);
241:     lv_btn_set_style(btn_run, LV_BTN_STYLE_REL, &toggle_btn_released);
242:     lv_btn_set_style(btn_run, LV_BTN_STYLE_PR, &toggle_btn_pressed);
243:     lv_btn_set_action(btn_run, LV_BTN_ACTION_CLICK, btn_run_action);
244:     lv_obj_set_width(btn_run, 150);
245:     lv_obj_set_height(btn_run, 25);
246:
247:     //label
248:     btn_run_label = lv_label_create(btn_run, NULL);
249:     lv_obj_set_style(btn_run_label, &heading_text);
250:     lv_label_set_text(btn_run_label, "Run Unit Test");
251:
252:
253:   //set up keyboard
254:   /*    keyboard = lv_kb_create(main_screen, NULL);
255:     lv_kb_set_mode(keyboard, LV_KB_MODE_NUM);
256:     lv_kb_set_ta(keyboard, kp_text_area);
257:     lv_kb_set_ta(keyboard, ki_text_area);
258:     lv_kb_set_ta(keyboard, kd_text_area);
259:     lv_kb_set_ta(keyboard, I_max_text_area);
260:     lv_kb_set_ta(keyboard, slew_text_area);
261:     lv_kb_set_ta(keyboard, port_text_area);*/
262:
263:
264:   //set positions
265:   //title
266:     lv_obj_set_pos(title_label, 100, 5);
267:
268:   //bottom buttons
269:     lv_obj_set_pos(btn_back, 30, 210);
270:     lv_obj_set_pos(btn_run, 300, 210);
271:
272:   //parameters side 1
273:     lv_obj_set_pos(kp_label, 20, 30);
274:     lv_obj_set_pos(ki_label, 20, 55);
275:     lv_obj_set_pos(kd_label, 20, 80);
276:     lv_obj_set_pos(I_max_label, 20, 105);
277:     lv_obj_set_pos(slew_label, 20, 130);
278:     lv_obj_set_pos(setpoint_label, 20, 155);
279:     lv_obj_set_pos(duration_label, 20, 180);
280:
281:     lv_obj_set_pos(kp_text_area, 130, 23);
282:     lv_obj_set_pos(ki_text_area, 130, 48);
283:     lv_obj_set_pos(kd_text_area, 130, 73);
284:     lv_obj_set_pos(I_max_text_area, 130, 98);
285:     lv_obj_set_pos(slew_text_area, 130, 123);
286:     lv_obj_set_pos(setpoint_text_area, 130, 148);
287:     lv_obj_set_pos(duration_text_area, 130, 173);
288:
289:   //parameters side 2
290:     lv_obj_set_pos(port_label, 240, 40);
291:     lv_obj_set_pos(gearset_label, 240, 75);
292:     lv_obj_set_pos(brakemode_label, 240, 100);
293:
294:     lv_obj_set_pos(port_text_area, 350, 33);
295:     lv_obj_set_pos(ddlist_gearset, 350, 75);
296:     lv_obj_set_pos(ddlist_brake_mode, 350, 100);
297:
298:   //information
299:     lv_obj_set_pos(information_label, 240, 140);
300:
301:
302:
303:   }
304:
305:   InternalMotorDebug::~InternalMotorDebug()
306:   {
307:     MotorThread* motor_thread = MotorThread::get_instance();
308:     motor_thread->unregister_motor(motor);
309:
310:     lv_obj_del(main_screen);
311:   }
312:
313:
314:   /**
315:    * sets cont to false signifying user wants to go back, main loop will exit
316:    */
317:   lv_res_t InternalMotorDebug::btn_back_action(lv_obj_t *btn)
318:   {
319:     cont = false;
320:     return LV_RES_OK;
321:   }
322:
323:   lv_res_t InternalMotorDebug::btn_run_action(lv_obj_t *btn)
324:   {
325:     lv_btn_set_state(btn, LV_BTN_STATE_INA);
326:     run = true;
327:     return LV_RES_OK;
328:   }
329:
330:
331:
332:   /**
333:    * looks at the string of the current drop down list option and compares it to
334:    * a string to see what gearset the user wants
335:    * sets gearset to this value
336:    */
337:   lv_res_t InternalMotorDebug::ddlist_gearset_action(lv_obj_t * ddlist)
338:   {
339:     //checks what the drop down list string is
```

```
340:     char sel_cstr[32];
341:     lv_ddlist_get_selected_str(ddlist, sel_cstr);
342:
343:     std::string sel_str = std::string(sel_cstr);  //convert to std::string so
344:                                                   //that the strings can be
345:                                                   //compared
346:
347:     //sets brake mode for motor
348:     if ( sel_str == "100" )
349:     {
350:         current_gearset = pros::E_MOTOR_GEARSET_36;
351:     }
352:
353:     else if ( sel_str == "600" )
354:     {
355:         current_gearset = pros::E_MOTOR_GEARSET_06;
356:     }
357:
358:     else
359:     {
360:         current_gearset = pros::E_MOTOR_GEARSET_18;
361:     }
362:
363:     return LV_RES_OK; //Return OK because the drop down list was not deleted
364: }
365:
366:
367:
368:
369: /**
370:  * looks at the string of the current drop down list option and compares it to
371:  * a string to see what brakemode the user wants
372:  * sets brake mode to this value
373:  */
374: lv_res_t InternalMotorDebug::ddlist_brake_mode_action(lv_obj_t * ddlist)
375: {
376:     //checks what the drop down list string is
377:     char sel_cstr[32];
378:     lv_ddlist_get_selected_str(ddlist, sel_cstr);
379:
380:     std::string sel_str = std::string(sel_cstr);  //convert to std::string so
381:                                                   //that the strings can be
382:                                                   //compared
383:
384:     //sets brake mode for motor
385:     if ( sel_str == "PID Hold" )
386:     {
387:         current_brake_mode = pros::E_MOTOR_BRAKE_HOLD;
388:     }
389:
390:     else if ( sel_str == "Brake" )
391:     {
392:         current_brake_mode = pros::E_MOTOR_BRAKE_BRAKE;
393:     }
394:
395:     else
396:     {
397:         current_brake_mode = pros::E_MOTOR_BRAKE_COAST;
398:     }
399:
400:     return LV_RES_OK; //Return OK if the drop down list is not deleted
401: }
402:
403:
404:
405:
406: int InternalMotorDebug::run_unit_test()
407: {
408:     pid pid_constants;
409:
410:     int slew = 0;
411:     int setpoint = 0;
412:     int motor_port = 0;
413:
414:     int duration = 0;
415:
416:     //read info from text areas in exception safe way
417:     try
418:     {
419:         double kP = std::stod(lv_ta_get_text(kp_text_area));
420:         double kI = std::stod(lv_ta_get_text(ki_text_area));
421:         double kD = std::stod(lv_ta_get_text(kd_text_area));
422:         double I_max = std::stod(lv_ta_get_text(I_max_text_area));
423:
424:         pid_constants.kP = kP;
425:         pid_constants.kP = kI;
426:         pid_constants.kP = kD;
427:         pid_constants.kP = I_max;
428:     }
429:     catch ( const std::invalid_argument& )
430:     {
431:         run = false;
432:         std::cerr << "[ERROR] " << pros::millis() << " invalid pid constants given to internal motor unit test\n";
433:         return 0;
434:     }
435:
436:     try
437:     {
438:         slew = std::stoi(lv_ta_get_text(slew_text_area));
439:     }
440:     catch ( const std::invalid_argument& )
441:     {
442:         run = false;
443:         std::cerr << "[ERROR] " << pros::millis() << " invalid slew rate given to internal motor unit test\n";
444:         return 0;
445:     }
446:
447:     try
448:     {
449:         setpoint = std::stoi(lv_ta_get_text(setpoint_text_area));
450:     }
451:     catch ( const std::invalid_argument& )
452:     {
```

```
453:         run = false;
454:         std::cerr << "[ERROR] " << pros::millis() << " invalid setpoint given to internal motor unit test\n";
455:         return 0;
456:     }
457:
458:     try
459:     {
460:         motor_port = std::stoi(lv_ta_get_text(port_text_area));
461:     }
462:     catch ( const std::invalid_argument& )
463:     {
464:         run = false;
465:         std::cerr << "[ERROR] " << pros::millis() << " invalid motor port given to internal motor unit test\n";
466:         return 0;
467:     }
468:
469:     try
470:     {
471:         duration = std::stoi(lv_ta_get_text(duration_text_area));
472:     }
473:     catch ( const std::invalid_argument& )
474:     {
475:         run = false;
476:         std::cerr << "[ERROR] " << pros::millis() << " invalid duration given to internal motor unit test\n";
477:         return 0;
478:     }
479:
480:     //set motor information
481:     motor.set_port(motor_port);
482:     motor.set_gearing(current_gearset);
483:     motor.set_brake_mode(current_brake_mode);
484:     if ( std::abs(slew) > 0 )
485:     {
486:         motor.enable_slew();
487:         motor.set_slew(slew);
488:     }
489:     else
490:     {
491:         motor.disable_slew();
492:     }
493:
494:     //motor.disable_velocity_pid();
495:     motor.disable_driver_control();
496:     motor.set_pid( pid_constants );
497:     motor.set_log_level(1);
498:
499:     int ut_end_time = pros::millis() + duration;
500:     motor.move_velocity(setpoint);
501:
502:     //wait for unit test to finish and update gui in the meantime
503:     while ( pros::millis() < ut_end_time )
504:     {
505:         //update gui
506:         std::string info_str;
507:         info_str = "Voltage: " + std::to_string(motor.get_actual_voltage()) + "\n";
508:         info_str += "Velocity: " + std::to_string(motor.get_actual_velocity()) + "\n";
509:         info_str += "Error: " + std::to_string(setpoint - motor.get_actual_velocity());
510:
511:         lv_label_set_text(information_label, info_str.c_str());
512:         pros::delay(50);
513:     }
514:
515:     motor.set_voltage(0);
516:
517:     return 1;
518: }
519:
520:
521:
522:
523: /**
524:  * waits for cont to be false which occurs when the user hits the back button
525:  */
526: void InternalMotorDebug::debug()
527: {
528:     std::string error_str = "-";
529:     cont = true;
530:     run = false;
531:
532:
533:     lv_scr_load(main_screen);
534:
535:     while ( cont )
536:     {
537:         //update information label
538:         std::string info_str;
539:         info_str = "Voltage: " + std::to_string(motor.get_actual_voltage()) + "\n";
540:         info_str += "Velocity: " + std::to_string(motor.get_actual_velocity()) + "\n";
541:         info_str += "Error: -";
542:         lv_label_set_text(information_label, info_str.c_str());
543:
544:         if ( run )
545:         {
546:             run_unit_test();
547:             run = false;
548:             lv_btn_set_state(btn_run, LV_BTN_STYLE_REL);
549:
550:         }
551:
552:
553:         pros::delay(100);
554:     }
555:
556:     motor.move(0);
557:     motor.set_log_level(0);
558: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/MotorsDebug.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/16/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * contains class that loads tabs to debug motors
8:   */
9:
10:  #ifndef __MOTORDEBUG_HPP__
11:  #define __MOTORDEBUG_HPP__
12:
13:  #include <vector>
14:
15:  #include "../../../../include/main.h"
16:
17:  #include "../Styles.hpp"
18:  #include "../../motors/Motors.hpp"
19:
20:
21:  //user defines
22:
23:  //sets size of container
24:  #define MOTORS_CONTAINER_WIDTH 440
25:  #define MOTORS_CONTAINER_HEIGHT 100
26:
27:  //sets percent at which to step velocity at
28:  //10 is reasonable because anything higher gives
29:  //less control and anything lower will make it
30:  //difficult to ramp up or down
31:  #define STEP_PERCENT 10
32:
33:
34:
35:  /**
36:   * @see: ../Styles.hpp
37:   *
38:   * general tab for one or two motors max
39:   * contains methods to show data and set velocity of motors
40:   * on this tab
41:   */
42:  class MotorsDebugTab : virtual Styles
43:  {
44:      private:
45:          lv_obj_t *container;
46:          lv_obj_t *motor1_label;
47:          lv_obj_t *motor2_label;
48:
49:          lv_obj_t *motor1_info;
50:          lv_obj_t *motor2_info;
51:
52:          std::vector<pros::Motor*> motors;
53:          std::vector<std::string> titles;
54:
55:      public:
56:          MotorsDebugTab( std::vector<pros::Motor*> motors_vec, std::vector<std::string> titles_vec, lv_obj_t *parent);
57:          ~MotorsDebugTab();
58:
59:          /**
60:           * @param: int target_velocity -> velocity the motor should be set to
61:           * @param: lv_obj_t* velocity_label -> label that current veolicty will be written to
62:           * @return: None
63:           *
64:           * @see: ../Styles.hpp
65:           * @see: ../../motors/Motors.hpp
66:           *
67:           * updates text for the motors that the class was instatiated with
68:           * also sets the velocity of the motor to int target_velocity
69:           * data shown is current drawn, voltage, reversed or not, temperature, encoder value,
70:           * and tourque
71:           */
72:          void update_label(int target_velocity, lv_obj_t *velocity_label);
73:  };
74:
75:
76:
77:
78:  /**
79:   * @see: class MotorsDebugTab
80:   * @see: ../Styles.hpp
81:   *
82:   * contatins debugger for motors
83:   * gives data for each motor set ie. left chassis, right chassis, intake, etc.
84:   */
85:  class MotorsDebug : virtual Styles
86:  {
87:      private:
88:          //screen
89:          lv_obj_t *motor_debug_screen;
90:
91:          //title label
92:          lv_obj_t *title_label;
93:
94:          //back button
95:          lv_obj_t *btn_back;
96:          lv_obj_t *btn_back_label;
97:
98:          /**
99:           * @param: lv_obj_t* btn -> button that called the funtion
100:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
101:          *
102:          * button callback function used to set cont to false meaning the
103:          * user wants to go to the title screen
104:          */
105:         static lv_res_t btn_back_action(lv_obj_t *btn);
106:
107:
108:         static lv_obj_t *tabview;   //tabview object
109:
110:         lv_obj_t *l_chassis_tab;    //individual tabs
111:         lv_obj_t *r_chassis_tab;    //content will come from base classes
112:         lv_obj_t *tilter_tab;
113:         lv_obj_t *intake_tab;
```

```
114:        lv_obj_t *lift_tab;
115:
116:        static uint16_t tab_loaded;  // 0 = left chassis
117:                                     // 1 = right chassis
118:                                     // 2 = lift
119:                                     // 3 = intake
120:                                     // 4 = lift
121:
122:        /**
123:         * @param: lv_obj_t* tabview -> tabview object for callback function
124:         * @param: uint16_t -> id of active tab
125:         * @return: lv_res_t -> return LV_RES_OK since object was not deleted
126:         *
127:         * funtion to stop motor movements and set the ability for other threads
128:         * to limit the speed of the motor ie. set it to zero in driver control
129:         * also updates target velocity and the tab loaded
130:         */
131:        static lv_res_t tab_load_action(lv_obj_t *tabview, uint16_t act_id);
132:
133:
134:        //velocity setting buttons
135:        lv_obj_t *velocity_label;
136:
137:        lv_obj_t *btn_pos_increase;
138:        lv_obj_t *btn_neg_increase;
139:        lv_obj_t *btn_stp;
140:
141:        lv_obj_t *btn_pos_increase_label;
142:        lv_obj_t *btn_neg_increase_label;
143:        lv_obj_t *btn_stp_label;
144:
145:
146:        /**
147:         * @param: lv_obj_t* btn -> button that called the funtion
148:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
149:         *
150:         * @see: std::tuple<int, int> get_velocity_step()
151:         *
152:         * button callback function used to decrease the target velocity
153:         */
154:        static lv_res_t btn_pos_increase_action(lv_obj_t *btn);
155:
156:
157:        /**
158:         * @param: lv_obj_t* btn -> button that called the funtion
159:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
160:         *
161:         * @see: std::tuple<int, int> get_velocity_step()
162:         *
163:         * button callback function used to increase the target velocity
164:         */
165:        static lv_res_t btn_neg_increase_action(lv_obj_t *btn);
166:
167:
168:        /**
169:         * @param: lv_obj_t* btn -> button that called the funtion
170:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
171:         *
172:         * @see: std::tuple<int, int> get_velocity_step()
173:         *
174:         * button callback function used to set target velocity to zero
175:         */
176:        static lv_res_t btn_stp_action(lv_obj_t *btn);
177:
178:
179:        /**
180:         * @return: std::tuple<int, int> -> tuple of step, a percentage of max velocity
181:         *          based on STEP_PERCENT, and max velocity of the motor
182:         * TOOD: update max velocity for motors and make more adaptable to changing motors
183:         *
184:         * gets the amount the step should be and the max velocity for the motor
185:         * the max velocity is higher than actual because the motor can go faster
186:         * than the specified RPM
187:         */
188:        static std::tuple<int, int> get_velocity_step();
189:
190:
191:        //static vars to help keep velocity
192:        //need to be static because they will be modified by
193:        //static function
194:        static int target_velocity;
195:
196:
197:        //brake mode option widgets
198:        lv_obj_t *brake_mode_label;
199:        lv_obj_t *ddlist_brake_mode;
200:
201:
202:        /**
203:         * @param: lv_obj_t* ddlist -> the dropdown list object for the callback function
204:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
205:         *
206:         * sets the brake mode for the motor set which will be updated in the main loop
207:         */
208:        static lv_res_t ddlist_brake_mode_action(lv_obj_t *ddlist);
209:
210:        static pros::motor_brake_mode_e_t current_brake_mode;
211:
212:    public:
213:        MotorsDebug();
214:        ~MotorsDebug();
215:
216:        static bool cont; //checks whether to keep letting user
217:                          //cycle through tabs
218:
219:        /**
220:         * @return: None
221:         *
222:         * allows user to interact with tabs for each motor set that display
223:         * data about those motors
224:         */
225:        void debug();
226:
```

```
227:  };
228:
229:
230:  #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/MotorsDebug.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/16/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: MotorsDebug.hpp
8:   *
9:   * contains classes and methods implementation that allow the gui to show
10:  * the user information about groups of motors seperated into tabs
11:  */
12:
13: #include <cstdint>
14: #include <cmath>
15: #include <vector>
16:
17: #include "../../../../include/main.h"
18: #include "../../../../include/api.h"
19:
20: #include "../Styles.hpp"
21: #include "../../motors/Motors.hpp"
22: #include "MotorsDebug.hpp"
23:
24:
25: //declare static members of all classes
26: bool MotorsDebug::cont = true;
27: int MotorsDebug::target_velocity = 0;
28: lv_obj_t *MotorsDebug::tabview;
29: pros::motor_brake_mode_e_t MotorsDebug::current_brake_mode = pros::E_MOTOR_BRAKE_COAST;
30: uint16_t MotorsDebug::tab_loaded = 0;
31:
32:
33:
34: MotorsDebugTab::MotorsDebugTab( std::vector<pros::Motor*> motors_vec, std::vector<std::string> titles_vec, lv_obj_t *parent)
35: {
36:     for( int i = 0; i < motors_vec.size(); i++ )
37:     {
38:         motors.push_back(motors_vec.at(i));
39:         titles.push_back(titles_vec.at(i));
40:     }
41:  //init container
42:     container = lv_cont_create(parent, NULL);
43:     lv_cont_set_fit(container, false, false);
44:     lv_obj_set_style(container, &gray);
45:     lv_cont_set_fit(container, false, false);
46:     lv_obj_set_width(container, MOTORS_CONTAINER_WIDTH);
47:     lv_obj_set_height(container, MOTORS_CONTAINER_HEIGHT);
48:
49:  //init motor 1 label
50:     motor1_label = lv_label_create(container, NULL);
51:     lv_obj_set_style(motor1_label, &toggle_tabbtn_pressed);
52:     lv_obj_set_width(motor1_label, (MOTORS_CONTAINER_WIDTH/2));
53:     lv_obj_set_height(motor1_label, 20);
54:     lv_label_set_align(motor1_label, LV_LABEL_ALIGN_CENTER);
55:     lv_label_set_text(motor1_label, titles.at(0).c_str());
56:
57:  //init motor 1 info label
58:     motor1_info = lv_label_create(container, NULL);
59:     lv_obj_set_style(motor1_info, &toggle_tabbtn_pressed);
60:     lv_obj_set_width(motor1_info, (MOTORS_CONTAINER_WIDTH/2));
61:     lv_obj_set_height(motor1_info, 20);
62:     lv_label_set_align(motor1_info, LV_LABEL_ALIGN_LEFT);
63:     lv_label_set_text(motor1_info, "None\nNone\nNone\nNone\nNone\nNone");
64:
65:  //init motor 2 label
66:     if( motors.size() > 1 )
67:     {
68:         motor2_label = lv_label_create(container, NULL);
69:         lv_obj_set_style(motor2_label, &toggle_tabbtn_pressed);
70:         lv_obj_set_width(motor2_label, (MOTORS_CONTAINER_WIDTH/2));
71:         lv_obj_set_height(motor2_label, 20);
72:         lv_label_set_align(motor2_label, LV_LABEL_ALIGN_CENTER);
73:         lv_label_set_text(motor2_label, titles.at(1).c_str());
74:
75:     //init motor 2 info label
76:         motor2_info = lv_label_create(container, NULL);
77:         lv_obj_set_style(motor2_info, &toggle_tabbtn_pressed);
78:         lv_obj_set_width(motor2_info, (MOTORS_CONTAINER_WIDTH/2));
79:         lv_obj_set_height(motor2_info, 20);
80:         lv_label_set_align(motor2_info, LV_LABEL_ALIGN_LEFT);
81:         lv_label_set_text(motor2_info, "None");
82:     }
83:
84:  //align objects on container
85:     lv_obj_set_pos(motor1_label, 60, 0);
86:     lv_obj_set_pos(motor1_info, 10, 15);
87:     if( motors.size() > 1 )
88:     {
89:         lv_obj_set_pos(motor2_label, 315, 0);
90:         lv_obj_set_pos(motor2_info, 255, 15);
91:     }
92: }
93:
94:
95: /**
96:  * function to be called in main loop so that data about motors will be updated
97:  * sets velocity, updates data, and updates velocity label
98:  */
99: void MotorsDebugTab::update_label(int target_velocity, lv_obj_t *velocity_label)
100: {
101:     std::string info1 = "";
102:     std::string info2 = "";
103:
104:  //set velocity to move at
105:     std::int32_t vel = target_velocity;
106:     motors.at(0)->move_velocity(vel);
107:     if ( motors.size() > 1 )
108:     {
109:         motors.at(1)->move_velocity(vel);
110:     }
111:
112:  //info for first motor
113:     info1 += "Current Draw: " + std::to_string(motors.at(0)->get_current_draw()) + "\n";
```

```
114:     info1 += "Voltage (mV): " + std::to_string(motors.at(0)->get_voltage()) + "\n";
115:     info1 += "State: ";
116:     info1 += motors.at(0)->is_reversed() ? "reversed\n" : "not reversed\n";
117:     info1 += "Temperature: " + std::to_string(motors.at(0)->get_temperature()) + "\n";
118:     info1 += "Encoder Position: " + std::to_string(motors.at(0)->get_position()) + "\n";
119:     info1 += "Torque (Nm): " + std::to_string(motors.at(0)->get_torque()) + "\n";
120:
121:   //info for second motor if it exists
122:     if ( motors.size() > 1 )
123:     {
124:         info2 += "Current Draw: " + std::to_string(motors.at(1)->get_current_draw()) + "\n";
125:         info2 += "Voltage (mV): " + std::to_string(motors.at(1)->get_voltage()) + "\n";
126:         info2 += "State: ";
127:         info2 += motors.at(1)->is_reversed() ? "reversed\n" : "not reversed\n";
128:         info2 += "Temperature: " + std::to_string(motors.at(1)->get_temperature()) + "\n";
129:         info2 += "Encoder Position: " + std::to_string(motors.at(1)->get_position()) + "\n";
130:         info2 += "Torque (Nm): " + std::to_string(motors.at(1)->get_torque()) + "\n";
131:     }
132:   //info for velocity label
133:     std::string velocity;
134:     velocity += titles.at(0) + ": " + std::to_string(motors.at(0)->get_actual_velocity()) + "\n";
135:     if ( motors.size() > 1 )
136:     {
137:         velocity += titles.at(1) + ": " + std::to_string(motors.at(1)->get_actual_velocity());
138:     }
139:
140:   //set labels
141:     //casts info strings to c strings to make them compatible with lvgl
142:     lv_label_set_text(motor1_info, info1.c_str());
143:     if ( motors.size() > 1 )
144:     {
145:         lv_label_set_text(motor2_info, info2.c_str());
146:     }
147:     lv_label_set_text(velocity_label, velocity.c_str());
148:
149:   }
150:
151:   MotorsDebugTab::~MotorsDebugTab()
152:   {
153:
154:   }
155:
156:
157:
158:
159:   MotorsDebug::MotorsDebug()
160:   {
161:     //set default for statics
162:     cont = true;
163:     target_velocity = 0;
164:     tab_loaded = 0;
165:
166:   //init screen
167:     motor_debug_screen = lv_obj_create(NULL, NULL);
168:     lv_obj_set_style(motor_debug_screen, &gray);
169:
170:   //init title label
171:     title_label = lv_label_create(motor_debug_screen, NULL);
172:     lv_label_set_style(title_label, &heading_text);
173:     lv_obj_set_width(title_label, MOTORS_CONTAINER_WIDTH);
174:     lv_obj_set_height(title_label, 20);
175:     lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
176:     lv_label_set_text(title_label, "Motors - Debug");
177:
178:   //init tabview
179:     tabview = lv_tabview_create(motor_debug_screen, NULL);
180:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BG, &gray);
181:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BTN_REL, &toggle_tabbtn_released);
182:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BTN_PR, &toggle_tabbtn_pressed);
183:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_INDIC, &sw_indic);
184:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BTN_TGL_REL, &toggle_tabbtn_pressed);
185:     lv_tabview_set_tab_load_action(tabview, tab_load_action);
186:     lv_obj_set_width(tabview, MOTORS_CONTAINER_WIDTH);
187:     lv_obj_set_height(tabview, 200);
188:
189:   //init tabs
190:     l_chassis_tab = lv_tabview_add_tab(tabview, "Chassis (L)");
191:     r_chassis_tab = lv_tabview_add_tab(tabview, "Chassis (R)");
192:     tilter_tab = lv_tabview_add_tab(tabview, "Tilter");
193:     intake_tab = lv_tabview_add_tab(tabview, "Intake");
194:     lift_tab = lv_tabview_add_tab(tabview, "Lift");
195:
196:   //init back button
197:     //button
198:     btn_back = lv_btn_create(motor_debug_screen, NULL);
199:     lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
200:     lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
201:     lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
202:     lv_obj_set_width(btn_back, 75);
203:     lv_obj_set_height(btn_back, 25);
204:
205:     //label
206:     btn_back_label = lv_label_create(btn_back, NULL);
207:     lv_obj_set_style(btn_back_label, &heading_text);
208:     lv_label_set_text(btn_back_label, "Back");
209:
210:   //init velocity label
211:     velocity_label = lv_label_create(motor_debug_screen, NULL);
212:     lv_obj_set_style(velocity_label, &subheading_text);
213:     lv_obj_set_width(velocity_label, 100);
214:     lv_obj_set_height(velocity_label, 40);
215:     lv_label_set_align(velocity_label, LV_LABEL_ALIGN_LEFT);
216:     lv_label_set_text(velocity_label, "Velocity: ");
217:
218:   //init velocity increase button
219:     //button
220:     btn_pos_increase =  lv_btn_create(motor_debug_screen, NULL);
221:     lv_btn_set_style(btn_pos_increase, LV_BTN_STYLE_REL, &toggle_btn_released);
222:     lv_btn_set_style(btn_pos_increase, LV_BTN_STYLE_PR, &toggle_btn_pressed);
223:     lv_btn_set_action(btn_pos_increase, LV_BTN_ACTION_CLICK, btn_pos_increase_action);
224:     lv_obj_set_width(btn_pos_increase, 40);
225:     lv_obj_set_height(btn_pos_increase, 25);
226:
```

```
227:     //label
228:     btn_pos_increase_label = lv_label_create(btn_pos_increase, NULL);
229:     lv_obj_set_style(btn_pos_increase_label, &heading_text);
230:     lv_label_set_text(btn_pos_increase_label, SYMBOL_RIGHT);
231:
232:   //init velocity decrease button
233:     //button
234:     btn_neg_increase =  lv_btn_create(motor_debug_screen, NULL);
235:     lv_btn_set_style(btn_neg_increase, LV_BTN_STYLE_REL, &toggle_btn_released);
236:     lv_btn_set_style(btn_neg_increase, LV_BTN_STYLE_PR, &toggle_btn_pressed);
237:     lv_btn_set_action(btn_neg_increase, LV_BTN_ACTION_CLICK, btn_neg_increase_action);
238:     lv_obj_set_width(btn_neg_increase, 40);
239:     lv_obj_set_height(btn_neg_increase, 25);
240:
241:     //label
242:     btn_neg_increase_label = lv_label_create(btn_neg_increase, NULL);
243:     lv_obj_set_style(btn_neg_increase_label, &heading_text);
244:     lv_label_set_text(btn_neg_increase_label, SYMBOL_LEFT);
245:
246:   //init zero velocity button
247:     //button
248:     btn_stp =  lv_btn_create(motor_debug_screen, NULL);
249:     lv_btn_set_style(btn_stp, LV_BTN_STYLE_REL, &toggle_btn_released);
250:     lv_btn_set_style(btn_stp, LV_BTN_STYLE_PR, &toggle_btn_pressed);
251:     lv_btn_set_action(btn_stp, LV_BTN_ACTION_CLICK, btn_stp_action);
252:     lv_obj_set_width(btn_stp, 40);
253:     lv_obj_set_height(btn_stp, 25);
254:
255:     //label
256:     btn_stp_label = lv_label_create(btn_stp, NULL);
257:     lv_obj_set_style(btn_stp_label, &heading_text);
258:     lv_label_set_text(btn_stp_label, SYMBOL_STOP);
259:
260:   //init brake mode label
261:     brake_mode_label = lv_label_create(motor_debug_screen, NULL);
262:     lv_obj_set_style(brake_mode_label, &heading_text);
263:     lv_obj_set_width(brake_mode_label, 100);
264:     lv_obj_set_height(brake_mode_label, 20);
265:     lv_label_set_align(brake_mode_label, LV_LABEL_ALIGN_CENTER);
266:     lv_label_set_text(brake_mode_label, "Brakemode: ");
267:
268:   //init drop down list
269:     ddlist_brake_mode = lv_ddlist_create(motor_debug_screen, NULL);
270:     lv_ddlist_set_options(ddlist_brake_mode, "Coast\n"
271:                           "Brake\n"
272:                           "PID Hold");
273:     lv_obj_set_style(ddlist_brake_mode, &subheading_text);
274:     lv_obj_set_width(ddlist_brake_mode, 125);
275:     lv_obj_set_height(ddlist_brake_mode, 18);
276:     lv_ddlist_set_action(ddlist_brake_mode, ddlist_brake_mode_action);
277:
278:   //set positions
279:     lv_obj_set_pos(btn_back, 30, 210);
280:     lv_obj_set_pos(btn_pos_increase, 270, 210);
281:     lv_obj_set_pos(btn_stp, 220, 210);
282:     lv_obj_set_pos(btn_neg_increase, 170, 210);
283:
284:     lv_obj_set_pos(velocity_label, 330, 177);
285:
286:     lv_obj_set_pos(brake_mode_label, 60, 177);
287:     lv_obj_set_pos(ddlist_brake_mode, 170, 177);
288:
289:     lv_obj_set_pos(title_label, 180, 5);
290:
291:     lv_obj_set_pos(tabview, 20, 25);
292:
293:   }
294:
295:
296:   MotorsDebug::~MotorsDebug()
297:   {
298:     Motors *motors = Motors::get_instance();
299:
300:     //sets motors to off
301:     motors->frontLeft->move_velocity(0);
302:     motors->backLeft->move_velocity(0);
303:     motors->frontRight->move_velocity(0);
304:     motors->backRight->move_velocity(0);
305:     motors->right_intake->move_velocity(0);
306:     motors->left_intake->move_velocity(0);
307:     motors->tilter->move_velocity(0);
308:     motors->lift->move_velocity(0);
309:
310:     //allow motor to go to zero for driver control if it is not set
311:     //already
312:     motors->allow_left_chassis = true;
313:     motors->allow_right_chassis = true;
314:     motors->allow_intake = true;
315:     motors->allow_tilter = true;
316:     motors->allow_lift = true;
317:
318:     //deletes widgets instantiated by class
319:     lv_obj_del(title_label);
320:
321:     lv_obj_del(btn_back_label);
322:     lv_obj_del(btn_back);
323:
324:     lv_obj_del(l_chassis_tab);
325:     lv_obj_del(r_chassis_tab);
326:     lv_obj_del(tilter_tab);
327:     lv_obj_del(intake_tab);
328:     lv_obj_del(lift_tab);
329:     lv_obj_del(tabview);
330:
331:     lv_obj_del(velocity_label);
332:
333:     lv_obj_del(btn_pos_increase_label);
334:     lv_obj_del(btn_neg_increase_label);
335:     lv_obj_del(btn_stp_label);
336:     lv_obj_del(btn_pos_increase);
337:     lv_obj_del(btn_neg_increase);
338:     lv_obj_del(btn_stp);
339:
```

```
340:     lv_obj_del(brake_mode_label);
341:     lv_obj_del(ddlist_brake_mode);
342:
343:
344:     lv_obj_del(motor_debug_screen);
345:   }
346:
347:
348:   /**
349:    * set cont to false to break main loop
350:    */
351:   lv_res_t MotorsDebug::btn_back_action(lv_obj_t *btn)
352:   {
353:     cont = false;
354:     return LV_RES_OK;
355:   }
356:
357:
358:   /**
359:    * callback function for when a new tab is selected
360:    * used to set motor to default ie. brakemode, velocity
361:    */
362:   lv_res_t MotorsDebug::tab_load_action(lv_obj_t *tabview, uint16_t act_id)
363:   {
364:     Motors *motors = Motors::get_instance();
365:
366:     //loads a new tab and sets default values for velocity of motors
367:     //and if motor is allowed to hit zero velocity
368:
369:     tab_loaded = act_id;
370:     target_velocity = 0;
371:
372:     motors->frontLeft->move_velocity(0);
373:     motors->backLeft->move_velocity(0);
374:     motors->frontRight->move_velocity(0);
375:     motors->backRight->move_velocity(0);
376:
377:     motors->allow_left_chassis = true;
378:     motors->allow_right_chassis = true;
379:     motors->allow_intake = true;
380:     motors->allow_tilter = true;
381:     motors->allow_lift = true;
382:
383:     return LV_RES_OK;
384:
385:   }
386:
387:
388:
389:   /**
390:    * looks at the current tab loaded to decide on max velocity because the motor
391:    * can be determined from that
392:    * gets the step percent by looking at what STEP_PERCENT is defined as
393:    */
394:   std::tuple<int, int> MotorsDebug::get_velocity_step()
395:   {
396:     int index = tab_loaded;   // 0 = left chassis - 200RPM
397:                               // 1 = right chassis - 200RPM
398:                               // 2 = tilter - 100RPM
399:                               // 3 = intake - 100RPM
400:                               // 4 = lift - 100RPM
401:     int max;
402:
403:     if ( index == 0 || index == 1 )
404:     {
405:       max = 250;
406:     }
407:
408:     else if (index == 2 || index == 3 || index == 4)
409:     {
410:       max = 130;
411:     }
412:
413:     else
414:     {
415:       max = 650;
416:     }
417:
418:     int step = static_cast<int>(max / STEP_PERCENT);
419:     return std::make_tuple(step, max);
420:   }
421:
422:
423:   /**
424:    * increses velocity of motor by calling get_velocity_step but limits it to
425:    * the max velocity
426:    */
427:   lv_res_t MotorsDebug::btn_pos_increase_action(lv_obj_t *btn)
428:   {
429:     //increases velocity by user defined percent
430:     int step;
431:     int max;
432:
433:     std::tie(step, max) = get_velocity_step();
434:     if ( target_velocity < max )
435:     {
436:       target_velocity = target_velocity + step;
437:     }
438:
439:     return LV_RES_OK;
440:   }
441:
442:
443:   /**
444:    * decreases velocity of motor by calling get_velocity_step but limits it to
445:    * the max velocity in the negative direction
446:    */
447:   lv_res_t MotorsDebug::btn_neg_increase_action(lv_obj_t *btn)
448:   {
449:     //decreases velocity by user defined percent
450:     int step;
451:     int max;
452:
```

```
453:     std::tie(step, max) = get_velocity_step();
454:     if ( target_velocity > 0-max )
455:     {
456:         target_velocity = target_velocity - step;
457:     }
458:
459:     return LV_RES_OK;
460: }
461:
462:
463: /**
464:  * sets velocity of motor to zero, used so that user does not have to click
465:  * many times to stop the motor
466:  */
467: lv_res_t MotorsDebug::btn_stp_action(lv_obj_t *btn)
468: {
469:     target_velocity = 0;
470:     return LV_RES_OK;
471: }
472:
473:
474: /**
475:  * looks at the string of the current drop down list option and compares it to
476:  * a string to see what brakemode the user wants
477:  * sets brake mode to this value
478:  */
479: lv_res_t MotorsDebug::ddlist_brake_mode_action(lv_obj_t * ddlist)
480: {
481:     //checks what the drop down list string is
482:     char sel_cstr[32];
483:     lv_ddlist_get_selected_str(ddlist, sel_cstr);
484:
485:     std::string sel_str = std::string(sel_cstr);  //convert to std::string so
486:                                                   //that the strings can be
487:                                                   //compared
488:
489:     //sets brake mode for motor
490:     if ( sel_str == "PID Hold" )
491:     {
492:         current_brake_mode = pros::E_MOTOR_BRAKE_HOLD;
493:     }
494:
495:     else if ( sel_str == "Brake" )
496:     {
497:         current_brake_mode = pros::E_MOTOR_BRAKE_BRAKE;
498:     }
499:
500:     else
501:     {
502:         current_brake_mode = pros::E_MOTOR_BRAKE_COAST;
503:     }
504:
505:     return LV_RES_OK; //Return OK if the drop down list is not deleted
506: }
507:
508:
509:
510:
511: /**
512:  * has a main loop that updates internal data as user cycles through tabs
513:  * to keep data relevent and motors following the function they are supposed to
514:  * loads tabs for each motor set
515:  */
516: void MotorsDebug::debug()
517: {
518:     Motors *motors = Motors::get_instance();
519:
520:     //used to check if user wants to continue cycling through
521:     //tabs. Will be set to zero and loop will break if user hits
522:     //the back button
523:     cont = true;
524:
525:     lv_tabview_set_tab_act(tabview, 0, NULL);
526:     lv_scr_load(motor_debug_screen);
527:
528:     MotorsDebugTab l_chassis_tab_debug( {motors->frontLeft, motors->backLeft}, {"Front Left", "Back Left"}, l_chassis_tab );
529:     MotorsDebugTab r_chassis_tab_debug( {motors->frontRight, motors->backRight}, {"Front Right", "Back Right"}, r_chassis_tab );
530:     MotorsDebugTab intake_tab_debug( {motors->right_intake, motors->left_intake}, {"Right Intake", "Left Intake"}, intake_tab );
531:     MotorsDebugTab tilter_tab_debug( {motors->tilter}, {"Tilter"}, intake_tab );
532:     MotorsDebugTab lift_tab_debug( {motors->lift}, {"Lift"}, lift_tab );
533:
534:     while ( cont )
535:     {
536:
537:         switch ( tab_loaded ) //switches to tab user wants to go to
538:         {
539:             case 0:
540:                 l_chassis_tab_debug.update_label(target_velocity, velocity_label);
541:                 motors->allow_left_chassis = motors->frontLeft->get_target_velocity() != 0 ? false : true;
542:                 break;
543:             case 1:
544:                 r_chassis_tab_debug.update_label(target_velocity, velocity_label);
545:                 motors->allow_right_chassis = motors->frontRight->get_target_velocity() != 0 ? false : true;
546:                 break;
547:             case 2:
548:                 intake_tab_debug.update_label(target_velocity, velocity_label);
549:                 motors->allow_intake = motors->right_intake->get_target_velocity() != 0 ? false : true;
550:                 break;
551:             case 3:
552:                 tilter_tab_debug.update_label(target_velocity, velocity_label);
553:                 motors->allow_tilter = motors->tilter->get_target_velocity() != 0 ? false : true;
554:                 break;
555:             case 4:
556:                 lift_tab_debug.update_label(target_velocity, velocity_label);
557:                 motors->allow_lift = motors->lift->get_target_velocity() != 0 ? false : true;
558:                 break;
559:         }
560:
561:         motors->frontLeft->set_brake_mode(current_brake_mode);
562:         motors->backLeft->set_brake_mode(current_brake_mode);
563:         motors->frontRight->set_brake_mode(current_brake_mode);
564:         motors->backRight->set_brake_mode(current_brake_mode);
565:         motors->right_intake->set_brake_mode(current_brake_mode);
```

```
566:        motors->left_intake->set_brake_mode(current_brake_mode);
567:        motors->tilter->set_brake_mode(current_brake_mode);
568:        motors->lift->set_brake_mode(current_brake_mode);
569:
570:  /*      std::cout << current_brake_mode << "\n";
571:        std::cout << motors->frontLeft.get_brake_mode() << "\n";
572:        std::cout << motors->backLeft.get_brake_mode() << "\n";
573:        std::cout << motors->frontRight.get_brake_mode() << "\n";
574:        std::cout << motors->backRight.get_brake_mode() << "\n";
575:        std::cout << motors->right_intake.get_brake_mode() << "\n";
576:        std::cout << motors->left_intake.get_brake_mode() << "\n";
577:        std::cout << motors->tilter.get_brake_mode() << "\n";
578:        std::cout << motors->lift.get_brake_mode() << "\n\n\n";*/
579:
580:        pros::delay(200);
581:    }
582:
583:    //reallow motor to hit zero velocity for driver controll
584:    motors->allow_left_chassis = true;
585:    motors->allow_right_chassis = true;
586:    motors->allow_intake = true;
587:    motors->allow_tilter = true;
588:    motors->allow_lift = true;
589: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/SensorsDebug.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   * TODO: condense, there are several classes that could be combined so that their is not so many
7:   *
8:   * contains classes for tabs of the sensors debugger tab
9:   */
10:
11: #ifndef __SENSORDEBUG_HPP__
12: #define __SENSORDEBUG_HPP__
13:
14: #include <string>
15:
16: #include "../../../../include/main.h"
17:
18: #include "../Styles.hpp"
19: #include "../Gimmicks.hpp"
20: #include "../../motors/Motors.hpp"
21: #include "../../sensors/Sensors.hpp"
22:
23:
24: //user defines
25:
26: //sets size of container
27: #define SENSORS_CONTAINER_WIDTH 440
28: #define SENSORS_CONTAINER_HEIGHT 120
29:
30:
31: //Base classes
32: //base classes are the tabs that will be loaded by the derived class
33: //this makes it easy to add new tabs while keeping the amount that has
34: //to go in one class to a minimum, especially since lvgl is not light
35:
36:
37:
38: /**
39:  * @see: ../Styles.hpp
40:  *
41:  * shows tab of IMEs and allows user to tare encoders and see values
42:  */
43: class IMEsDebugger :
44:     virtual Styles,
45:     virtual Sensors
46: {
47:     private:
48:         lv_obj_t *container;
49:         lv_obj_t *title;
50:         lv_obj_t *info;
51:
52:         lv_obj_t *btn_tare;
53:         lv_obj_t *btn_tare_label;
54:
55:         /**
56:          * @param: lv_obj_t* btn -> button that called the funtion
57:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
58:          *
59:          * button callback function used to tare all IMEs
60:          */
61:         static lv_res_t btn_tare_action(lv_obj_t *btn);
62:
63:
64:     protected:
65:         /**
66:          * @return: None
67:          *
68:          * updates values for IMEs
69:          */
70:         void update_imes_info();
71:
72:     public:
73:         IMEsDebugger();
74:         virtual ~IMEsDebugger();
75:
76:         /**
77:          * @param: lv_obj_t* parent -> parent of the tab
78:          * @return: None
79:          *
80:          * objects are initially loaded onto a NULL parent to be updated later
81:          * this sets it so that the parent of the objects is now the tab
82:          */
83:         void IMEsDebuggerInit(lv_obj_t *parent);
84:
85: };
86:
87:
88:
89: /**
90:  * @see: ../Styles.
91:  *
92:  * show values for accelerometer
93:  */
94: class AccelerometerDebugger :
95:     virtual Styles,
96:     virtual Sensors
97: {
98:     private:
99:         lv_obj_t *container;
100:
101:         lv_obj_t *title1;
102:         lv_obj_t *title2;
103:         lv_obj_t *title3;
104:
105:         lv_obj_t *info1;
106:         lv_obj_t *info2;
107:         lv_obj_t *info3;
108:
109:         lv_obj_t *btn_calibrate;
110:         lv_obj_t *btn_calibrate_label;
111:
112:
113:         /**
```

```
114:         * @param: lv_obj_t* btn -> button that called the funtion
115:         * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
116:         *
117:         * button callback function used to calibrate sensor
118:         */
119:        static lv_res_t btn_calibrate_action(lv_obj_t *btn);
120:
121:    protected:
122:        /**
123:         * @return: None
124:         *
125:         * updates value of x, y, and z values of accelerometer
126:         */
127:        void update_accelerometer_info();
128:
129:    public:
130:        AccelerometerDebugger();
131:        virtual ~AccelerometerDebugger();
132:
133:        /**
134:         * @param: lv_obj_t* parent -> parent of the tab
135:         * @return: None
136:         *
137:         * objects are initially loaded onto a NULL parent to be updated later
138:         * this sets it so that the parent of the objects is now the tab
139:         */
140:        void AccelerometerDebuggerInit(lv_obj_t *parent);
141:  };
142:
143:
144:
145:  /**
146:   * @see: ../Styles.
147:   *
148:   * show values for gyro(s)
149:   */
150:  class GyrosDebugger :
151:      virtual Styles,
152:      virtual Sensors
153:  {
154:      private:
155:          lv_obj_t *container;
156:
157:          lv_obj_t *title1;
158:          lv_obj_t *title2;
159:          lv_obj_t *title3;
160:
161:          lv_obj_t *info1;
162:          lv_obj_t *info2;
163:          lv_obj_t *info3;
164:
165:          lv_obj_t *btn_calibrate;
166:          lv_obj_t *btn_calibrate_label;
167:
168:          /**
169:           * @param: lv_obj_t* btn -> button that called the funtion
170:           * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
171:           *
172:           * button callback function used to calibrate sensor
173:           */
174:          static lv_res_t btn_calibrate_action(lv_obj_t *btn);
175:
176:      protected:
177:          /**
178:           * @return: None
179:           *
180:           * updates values of gyro(s)
181:           */
182:          void update_gyro_info();
183:
184:      public:
185:          GyrosDebugger();
186:          virtual ~GyrosDebugger();
187:
188:          /**
189:           * @param: lv_obj_t* parent -> parent of the tab
190:           * @return: None
191:           *
192:           * objects are initially loaded onto a NULL parent to be updated later
193:           * this sets it so that the parent of the objects is now the tab
194:           */
195:          void GyrosDebuggerInit(lv_obj_t *parent);
196:  };
197:
198:
199:
200:  /**
201:   * @see: ../Styles.
202:   *
203:   * show value for potentiometer
204:   */
205:  class PotentiometerDebugger :
206:      virtual Styles,
207:      virtual Sensors
208:  {
209:      private:
210:          lv_obj_t *container;
211:
212:          lv_obj_t *title1;
213:          lv_obj_t *title2;
214:          lv_obj_t *title3;
215:
216:          lv_obj_t *info1;
217:          lv_obj_t *info2;
218:          lv_obj_t *info3;
219:
220:          lv_obj_t *btn_calibrate;
221:          lv_obj_t *btn_calibrate_label;
222:
223:          /**
224:           * @param: lv_obj_t* btn -> button that called the funtion
225:           * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
226:           *
```

```
227:         * button callback function used to calibrate sensor
228:         */
229:        static lv_res_t btn_calibrate_action(lv_obj_t *btn);
230:
231:    protected:
232:        /**
233:         * @return: None
234:         *
235:         * updates value of potentiometer
236:         */
237:        void update_pot_info();
238:
239:    public:
240:        PotentiometerDebugger();
241:        virtual ~PotentiometerDebugger();
242:
243:        /**
244:         * @param: lv_obj_t* parent -> parent of the tab
245:         * @return: None
246:         *
247:         * objects are initially loaded onto a NULL parent to be updated later
248:         * this sets it so that the parent of the objects is now the tab
249:         */
250:        void PotentiometerDebuggerInit(lv_obj_t *parent);
251:  };
252:
253:
254:
255:  /**
256:   * @see: ../Styles.
257:   *
258:   * show value for limit switch
259:   */
260:  class LimitSwitchDebugger :
261:      virtual Styles,
262:      virtual Sensors
263:  {
264:  private:
265:      lv_obj_t *container;
266:
267:      lv_obj_t *title1;
268:      lv_obj_t *title2;
269:
270:      lv_obj_t *info1;
271:      lv_obj_t *info2;
272:
273:      protected:
274:          /**
275:           * @return: None
276:           *
277:           * updates value of limit switch
278:           */
279:          void update_limit_switch_info();
280:
281:      public:
282:          LimitSwitchDebugger();
283:          virtual ~LimitSwitchDebugger();
284:
285:          /**
286:           * @param: lv_obj_t* parent -> parent of the tab
287:           * @return: None
288:           *
289:           * objects are initially loaded onto a NULL parent to be updated later
290:           * this sets it so that the parent of the objects is now the tab
291:           */
292:          void LimitSwitchDebuggerInit(lv_obj_t *parent);
293:  };
294:
295:
296:
297:  /**
298:   * @see: ../Styles.
299:   *
300:   * sets value for LED
301:   */
302:  class LEDDebugger :
303:      virtual Styles,
304:      virtual Sensors
305:  {
306:  private:
307:          lv_obj_t *container;
308:
309:          lv_obj_t *btn_set;
310:          lv_obj_t *btn_set_label;
311:
312:          /**
313:           * @param: lv_obj_t* btn -> button that called the funtion
314:           * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
315:           *
316:           * button callback function used to turn led on
317:           */
318:          static lv_res_t btn_set_action(lv_obj_t *btn);
319:
320:          lv_obj_t *btn_clear;
321:          lv_obj_t *btn_clear_label;
322:
323:          /**
324:           * @param: lv_obj_t* btn -> button that called the funtion
325:           * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
326:           *
327:           * button callback function used to turn led off
328:           */
329:          static lv_res_t btn_clear_action(lv_obj_t *btn);
330:
331:      public:
332:          LEDDebugger();
333:          virtual ~LEDDebugger();
334:
335:          /**
336:           * @param: lv_obj_t* parent -> parent of the tab
337:           * @return: None
338:           *
339:           * objects are initially loaded onto a NULL parent to be updated later
```

```
340:         * this sets it so that the parent of the objects is now the tab
341:         */
342:        void LEDDebuggerInit(lv_obj_t *parent);
343:    };
344:
345:
346:
347:    /**
348:     * @see: ../Styles.
349:     *
350:     * starts new page with debugger info for vision sensor because it needs more room
351:     */
352:    class VisionSensorDebugger : virtual Styles
353:    {
354:        private:
355:            lv_obj_t *title_label;
356:
357:            lv_obj_t *vision_sensor_screen;
358:
359:            //back button
360:            lv_obj_t *btn_back;
361:            lv_obj_t *btn_back_label;
362:
363:            /**
364:             * @param: lv_obj_t* btn -> button that called the funtion
365:             * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
366:             *
367:             * button callback function used to go back from the new screen loaded by
368:             * this tab because it is predicted to need more space
369:             */
370:            static lv_res_t btn_back_action(lv_obj_t *btn);
371:
372:            static bool cont;
373:
374:        protected:
375:            /**
376:             * @return: None
377:             *
378:             * loads a new page with debug info
379:             */
380:            void load_vision_sensor_page();
381:
382:        public:
383:            VisionSensorDebugger();
384:            virtual ~VisionSensorDebugger();
385:    };
386:
387:
388:
389:
390:    //derived class
391:
392:
393:    /**
394:     * @see: ../Styles.
395:     *
396:     * starts tab object with all the sensor tabs that the user
397:     * can switch between
398:     */
399:    class SensorsDebug :
400:        virtual private Styles,
401:        virtual private Sensors,
402:        private IMEsDebugger,
403:        private AccelerometerDebugger,
404:        private GyrosDebugger,
405:        private PotentiometerDebugger,
406:        private LimitSwitchDebugger,
407:        private LEDDebugger,
408:        private VisionSensorDebugger
409:    {
410:        private:
411:            //screen
412:            lv_obj_t *sensors_debug_screen;
413:
414:            //title label
415:            lv_obj_t *title_label;
416:
417:            //back button
418:            lv_obj_t *btn_back;
419:            lv_obj_t *btn_back_label;
420:
421:            /**
422:             * @param: lv_obj_t* btn -> button that called the funtion
423:             * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
424:             *
425:             * button callback function used to go back from the debug screen to
426:             * the title screen
427:             */
428:            static lv_res_t btn_back_action(lv_obj_t *btn);
429:
430:
431:            static lv_obj_t *tabview; //tabview object
432:
433:            lv_obj_t *imes_tab;   //individual tabs
434:            lv_obj_t *acclr_tab;  //content will come from base classes
435:            lv_obj_t *gyros_tab;
436:            lv_obj_t *pot_tab;
437:            lv_obj_t *limit_tab;
438:            lv_obj_t *led_tab;
439:            lv_obj_t *vision_sensor_tab;
440:
441:
442:        public:
443:            SensorsDebug();
444:            ~SensorsDebug();
445:
446:            static bool all_cont; //checks whether to allow user to
447:                                  //cycle through tabs or not
448:
449:
450:            /**
451:             * @return: None
452:             *
```

```
453:          * contains methods for transition between tabs with checking sensors
454:          * for if they are calibrated or not
455:          * waits for user to go back in a loop while also switching tabs
456:          *
457:          */
458:         void debug();
459:
460:   };
461:
462:
463:
464:
465:   #endif
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/SensorsDebug.cpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * @see: SensorsDebug.hpp
8:   *
9:   * contains all methods for tabs that contain ways to debug and check sensors
10:  */
11:
12:  #include "../../../../include/main.h"
13:  #include "../../../../include/api.h"
14:
15:  #include "../Styles.hpp"
16:  #include "../Gimmicks.hpp"
17:  #include "../../motors/Motors.hpp"
18:  #include "../../sensors/Sensors.hpp"
19:  #include "SensorsDebug.hpp"
20:
21:  //base classes
22:  bool VisionSensorDebugger::cont = true;
23:  bool SensorsDebug::all_cont = true;
24:  lv_obj_t *SensorsDebug::tabview;
25:
26:
27:  IMEsDebugger::IMEsDebugger()
28:  {
29:  //init container
30:      container = lv_cont_create(lv_scr_act(), NULL);
31:      lv_cont_set_fit(container, false, false);
32:      lv_obj_set_style(container, &gray);
33:      lv_cont_set_fit(container, false, false);
34:      lv_obj_set_width(container, SENSORS_CONTAINER_WIDTH);
35:      lv_obj_set_height(container, SENSORS_CONTAINER_HEIGHT);
36:
37:  //default text
38:      std::string text = (
39:          "front right  -\n"
40:          "back right   -\n"
41:          "front left   -\n"
42:          "back left    -\n"
43:          "right lift   -\n"
44:          "left lift    -\n"
45:          "intake       -\n"
46:          "lift         - "
47:      );
48:
49:  //init integrated motor encoders label label
50:      info = lv_label_create(container, NULL);
51:      lv_obj_set_style(info, &toggle_tabbtn_pressed);
52:      lv_obj_set_width(info, (SENSORS_CONTAINER_WIDTH));
53:      lv_obj_set_height(info, SENSORS_CONTAINER_HEIGHT);
54:      lv_label_set_align(info, LV_LABEL_ALIGN_LEFT);
55:      lv_label_set_text(info, text.c_str());
56:
57:
58:  //init tare encoders button
59:      //button
60:      btn_tare = lv_btn_create(container, NULL);
61:      lv_btn_set_style(btn_tare, LV_BTN_STYLE_REL, &toggle_btn_released);
62:      lv_btn_set_style(btn_tare, LV_BTN_STYLE_PR, &toggle_btn_pressed);
63:      lv_btn_set_action(btn_tare, LV_BTN_ACTION_CLICK, btn_tare_action);
64:      lv_obj_set_width(btn_tare, 110);
65:      lv_obj_set_height(btn_tare, 25);
66:
67:      //label
68:      btn_tare_label = lv_label_create(btn_tare, NULL);
69:      lv_obj_set_style(btn_tare_label, &subheading_text);
70:      lv_label_set_text(btn_tare_label, "tare encoders");
71:
72:
73:  //align objects on container
74:      lv_obj_set_pos(info, 10, 0);
75:      lv_obj_set_pos(btn_tare, 300, (SENSORS_CONTAINER_HEIGHT - 30));
76:  }
77:
78:  IMEsDebugger::~IMEsDebugger()
79:  {
80:
81:  }
82:
83:
84:  /**
85:   * tares encodes of all motors
86:   */
87:  lv_res_t IMEsDebugger::btn_tare_action(lv_obj_t *btn)
88:  {
89:      Motors *motors = Motors::get_instance();
90:
91:      motors->frontRight->tare_position();
92:      motors->backRight->tare_position();
93:      motors->frontLeft->tare_position();
94:      motors->backLeft->tare_position();
95:      motors->right_intake->tare_position();
96:      motors->left_intake->tare_position();
97:      motors->tilter->tare_position();
98:      motors->lift->tare_position();
99:  }
100:
101:
102: /**
103:  * changes parent of objects
104:  */
105: void IMEsDebugger::IMEsDebuggerInit(lv_obj_t *parent)
106: {
107:     //sets parent of container to pointer of new parent
108:     //this is to allow seperation of tabs into seperate classes
109:     //reduce the quantity in one class and to allow for ease of adding
110:     //new or different tabs
111:
112:     lv_obj_set_parent(container, parent);
113: }
```

```
114:
115:
116: /**
117:  * updates for each motor to current values
118:  */
119: void IMEsDebugger::update_imes_info()
120: {
121:     Motors *motors = Motors::get_instance();
122:
123:     std::string text = (
124:         "front right   - " + std::to_string(motors->frontRight->get_position()) + "\n"
125:         "back right    - " + std::to_string(motors->backRight->get_position()) + "\n"
126:         "front left    - " + std::to_string(motors->frontLeft->get_position()) + "\n"
127:         "back left     - " + std::to_string(motors->backLeft->get_position()) + "\n"
128:         "right intake  - " + std::to_string(motors->right_intake->get_position()) + "\n"
129:         "left intake   - " + std::to_string(motors->left_intake->get_position()) + "\n"
130:         "tilter        - " + std::to_string(motors->tilter->get_position()) + "\n"
131:         "lift          - " + std::to_string(motors->lift->get_position()) + "\n"
132:     );
133:
134:     lv_label_set_text(info, text.c_str());
135: }
136:
137:
138:
139:
140:
141: AccelerometerDebugger::AccelerometerDebugger()
142: {
143: //init container
144:     container = lv_cont_create(lv_scr_act(), NULL);
145:     lv_cont_set_fit(container, false, false);
146:     lv_obj_set_style(container, &gray);
147:     lv_cont_set_fit(container, false, false);
148:     lv_obj_set_width(container, SENSORS_CONTAINER_WIDTH);
149:     lv_obj_set_height(container, SENSORS_CONTAINER_HEIGHT);
150:
151: //title for columns
152:     //1
153:     title1 = lv_label_create(container, NULL);
154:     lv_obj_set_style(title1, &toggle_tabbtn_pressed);
155:     lv_obj_set_width(title1, (SENSORS_CONTAINER_WIDTH));
156:     lv_obj_set_height(title1, 20);
157:     lv_label_set_align(title1, LV_LABEL_ALIGN_CENTER);
158:     lv_label_set_text(title1, "None");
159:
160:     //2
161:     title2 = lv_label_create(container, NULL);
162:     lv_obj_set_style(title2, &toggle_tabbtn_pressed);
163:     lv_obj_set_width(title2, (SENSORS_CONTAINER_WIDTH/3));
164:     lv_obj_set_height(title2, 20);
165:     lv_label_set_align(title2, LV_LABEL_ALIGN_CENTER);
166:     lv_label_set_text(title2, "None");
167:
168:     //3
169:     title3 = lv_label_create(container, NULL);
170:     lv_obj_set_style(title3, &toggle_tabbtn_pressed);
171:     lv_obj_set_width(title3, (SENSORS_CONTAINER_WIDTH/3));
172:     lv_obj_set_height(title3, 20);
173:     lv_label_set_align(title3, LV_LABEL_ALIGN_CENTER);
174:     lv_label_set_text(title3, "None");
175:
176: //info for columns
177:     //1
178:     info1 = lv_label_create(container, NULL);
179:     lv_obj_set_style(info1, &toggle_tabbtn_pressed);
180:     lv_obj_set_width(info1, (SENSORS_CONTAINER_WIDTH/3));
181:     lv_obj_set_height(info1, SENSORS_CONTAINER_HEIGHT - 20);
182:     lv_label_set_align(info1, LV_LABEL_ALIGN_LEFT);
183:     lv_label_set_text(info1, "None");
184:
185:     //2
186:     info2 = lv_label_create(container, NULL);
187:     lv_obj_set_style(info2, &toggle_tabbtn_pressed);
188:     lv_obj_set_width(info2, (SENSORS_CONTAINER_WIDTH / 3));
189:     lv_obj_set_height(info2, SENSORS_CONTAINER_HEIGHT - 20);
190:     lv_label_set_align(info2, LV_LABEL_ALIGN_LEFT);
191:     lv_label_set_text(info2, "None");
192:
193:     //3
194:     info3 = lv_label_create(container, NULL);
195:     lv_obj_set_style(info3, &toggle_tabbtn_pressed);
196:     lv_obj_set_width(info3, (SENSORS_CONTAINER_WIDTH/3));
197:     lv_obj_set_height(info3, SENSORS_CONTAINER_HEIGHT - 20);
198:     lv_label_set_align(info3, LV_LABEL_ALIGN_LEFT);
199:     lv_label_set_text(info3, "None");
200:
201: //calibrate button
202:     //button
203:     btn_calibrate = lv_btn_create(container, NULL);
204:     lv_btn_set_style(btn_calibrate, LV_BTN_STYLE_REL, &toggle_btn_released);
205:     lv_btn_set_style(btn_calibrate, LV_BTN_STYLE_PR, &toggle_btn_pressed);
206:     lv_btn_set_action(btn_calibrate, LV_BTN_ACTION_CLICK, btn_calibrate_action);
207:     lv_obj_set_width(btn_calibrate, 110);
208:     lv_obj_set_height(btn_calibrate, 25);
209:
210:     //label
211:     btn_calibrate_label = lv_label_create(btn_calibrate, NULL);
212:     lv_obj_set_style(btn_calibrate_label, &subheading_text);
213:     lv_label_set_text(btn_calibrate_label, "Calibrate");
214:
215: //set positions relative to container
216:     lv_obj_align(title1, container, LV_ALIGN_IN_TOP_LEFT, 10, 10);
217:     lv_obj_align(info1, container, LV_ALIGN_IN_TOP_LEFT, 10, 30);
218:
219:     lv_obj_align(title2, container, LV_ALIGN_IN_TOP_MID, -15, 10);
220:     lv_obj_align(info2, container, LV_ALIGN_IN_TOP_MID, -15, 30);
221:
222:     lv_obj_align(title3, container, LV_ALIGN_IN_TOP_RIGHT, -100, 10);
223:     lv_obj_align(info3, container, LV_ALIGN_IN_TOP_RIGHT, -100, 30);
224:
225:     lv_obj_align(btn_calibrate, container, LV_ALIGN_IN_BOTTOM_RIGHT, -50, 0);
226: }
```

```
227:
228:
229: AccelerometerDebugger::~AccelerometerDebugger()
230: {
231:
232: }
233:
234:
235: /**
236:  * calibrates accelerometer and adds a loading bar so the gui doesn't appear to
237:  * hang for no reason
238:  */
239: lv_res_t AccelerometerDebugger::btn_calibrate_action(lv_obj_t *btn)
240: {
241:     Loading load;
242:     load.show_load(1800, lv_scr_act(), 190, 240); //shows loading bar while calibrating
243:     Sensors::calibrateAccel();
244:     load.hide_load();
245:
246:     return LV_RES_OK;
247: }
248:
249:
250: /**
251:  * changes parent of objects
252:  */
253: void AccelerometerDebugger::AccelerometerDebuggerInit(lv_obj_t *parent)
254: {
255:     //sets parent of container to pointer of new parent
256:     //this is to allow seperation of tabs into seperate classes
257:     //reduce the quantity in one class and to allow for ease of adding
258:     //new or different tabs
259:
260:     lv_obj_set_parent(container, parent);
261: }
262:
263:
264: /**
265:  * updates data for each axis, shows raw data and corrected
266:  */
267: void AccelerometerDebugger::update_accelerometer_info()
268: {
269:     std::string names_title = "Accelerometer Axis";
270:     std::string raw_title = "Raw Input";
271:     std::string corrected_title = "Corrected Input";
272:
273:     std::string names = "X Axis\nY Axis\nZ Axis";
274:
275:     accelValues raw_vals = getRawAccelerometer();
276:     accelValues corrected_vals = getCorrectedAccelerometer();
277:
278:     std::string raw = (
279:         std::to_string(raw_vals.Xaxis) + "\n" +
280:         std::to_string(raw_vals.Yaxis) + "\n" +
281:         std::to_string(raw_vals.Zaxis) + "\n"
282:     );
283:
284:     std::string corrected = (
285:         std::to_string(corrected_vals.Xaxis) + "\n" +
286:         std::to_string(corrected_vals.Yaxis) + "\n" +
287:         std::to_string(corrected_vals.Zaxis) + "\n"
288:     );
289:
290:     lv_label_set_text(title1, names_title.c_str());
291:     lv_label_set_text(title2, raw_title.c_str());
292:     lv_label_set_text(title3, corrected_title.c_str());
293:     lv_label_set_text(info1, names.c_str());
294:     lv_label_set_text(info2, raw.c_str());
295:     lv_label_set_text(info3, corrected.c_str());
296:
297:
298: }
299:
300:
301:
302:
303:
304: GyrosDebugger::GyrosDebugger()
305: {
306: //init container
307:     container = lv_cont_create(lv_scr_act(), NULL);
308:     lv_cont_set_fit(container, false, false);
309:     lv_obj_set_style(container, &gray);
310:     lv_cont_set_fit(container, false, false);
311:     lv_obj_set_width(container, SENSORS_CONTAINER_WIDTH);
312:     lv_obj_set_height(container, SENSORS_CONTAINER_HEIGHT);
313:
314: //title for columns
315:     //1
316:     title1 = lv_label_create(container, NULL);
317:     lv_obj_set_style(title1, &toggle_tabbtn_pressed);
318:     lv_obj_set_width(title1, (SENSORS_CONTAINER_WIDTH));
319:     lv_obj_set_height(title1, 20);
320:     lv_label_set_align(title1, LV_LABEL_ALIGN_CENTER);
321:     lv_label_set_text(title1, "None");
322:
323:     //2
324:     title2 = lv_label_create(container, NULL);
325:     lv_obj_set_style(title2, &toggle_tabbtn_pressed);
326:     lv_obj_set_width(title2, (SENSORS_CONTAINER_WIDTH/3));
327:     lv_obj_set_height(title2, 20);
328:     lv_label_set_align(title2, LV_LABEL_ALIGN_CENTER);
329:     lv_label_set_text(title2, "None");
330:
331:     //3
332:     title3 = lv_label_create(container, NULL);
333:     lv_obj_set_style(title3, &toggle_tabbtn_pressed);
334:     lv_obj_set_width(title3, (SENSORS_CONTAINER_WIDTH/3));
335:     lv_obj_set_height(title3, 20);
336:     lv_label_set_align(title3, LV_LABEL_ALIGN_CENTER);
337:     lv_label_set_text(title3, "None");
338:
339: //info for columns
```

```
340:     //1
341:     info1 = lv_label_create(container, NULL);
342:     lv_obj_set_style(info1, &toggle_tabbtn_pressed);
343:     lv_obj_set_width(info1, (SENSORS_CONTAINER_WIDTH/3));
344:     lv_obj_set_height(info1, SENSORS_CONTAINER_HEIGHT - 20);
345:     lv_label_set_align(info1, LV_LABEL_ALIGN_LEFT);
346:     lv_label_set_text(info1, "None");
347:
348:     //2
349:     info2 = lv_label_create(container, NULL);
350:     lv_obj_set_style(info2, &toggle_tabbtn_pressed);
351:     lv_obj_set_width(info2, (SENSORS_CONTAINER_WIDTH / 3));
352:     lv_obj_set_height(info2, SENSORS_CONTAINER_HEIGHT - 20);
353:     lv_label_set_align(info2, LV_LABEL_ALIGN_LEFT);
354:     lv_label_set_text(info2, "None");
355:
356:     //3
357:     info3 = lv_label_create(container, NULL);
358:     lv_obj_set_style(info3, &toggle_tabbtn_pressed);
359:     lv_obj_set_width(info3, (SENSORS_CONTAINER_WIDTH/3));
360:     lv_obj_set_height(info3, SENSORS_CONTAINER_HEIGHT - 20);
361:     lv_label_set_align(info3, LV_LABEL_ALIGN_LEFT);
362:     lv_label_set_text(info3, "None");
363:
364:   //calibrate button
365:     //button
366:     btn_calibrate = lv_btn_create(container, NULL);
367:     lv_btn_set_style(btn_calibrate, LV_BTN_STYLE_REL, &toggle_btn_released);
368:     lv_btn_set_style(btn_calibrate, LV_BTN_STYLE_PR, &toggle_btn_pressed);
369:     lv_btn_set_action(btn_calibrate, LV_BTN_ACTION_CLICK, btn_calibrate_action);
370:     lv_obj_set_width(btn_calibrate, 110);
371:     lv_obj_set_height(btn_calibrate, 25);
372:
373:     //label
374:     btn_calibrate_label = lv_label_create(btn_calibrate, NULL);
375:     lv_obj_set_style(btn_calibrate_label, &subheading_text);
376:     lv_label_set_text(btn_calibrate_label, "Calibrate");
377:
378:   //set positions relative to container
379:     lv_obj_align(title1, container, LV_ALIGN_IN_TOP_LEFT, 10, 10);
380:     lv_obj_align(info1, container, LV_ALIGN_IN_TOP_LEFT, 10, 30);
381:
382:     lv_obj_align(title2, container, LV_ALIGN_IN_TOP_MID, -15, 10);
383:     lv_obj_align(info2, container, LV_ALIGN_IN_TOP_MID, -15, 30);
384:
385:     lv_obj_align(title3, container, LV_ALIGN_IN_TOP_RIGHT, -100, 10);
386:     lv_obj_align(info3, container, LV_ALIGN_IN_TOP_RIGHT, -100, 30);
387:
388:     lv_obj_align(btn_calibrate, container, LV_ALIGN_IN_BOTTOM_RIGHT, -50, 0);
389:   }
390:
391:   GyrosDebugger::~GyrosDebugger()
392:   {
393:
394:   }
395:
396:
397:   /**
398:    * calibrates gyro(s) and adds loading bar so gui doesn't appear to hang for no reason
399:    */
400:   lv_res_t GyrosDebugger::btn_calibrate_action(lv_obj_t *btn)
401:   {
402:     Loading load;
403:     load.show_load(1000, lv_scr_act(), 190, 240); //shows loading bar while calibrating
404:     Sensors::calibrateGyro();
405:     load.hide_load();
406:
407:     return LV_RES_OK;
408:   }
409:
410:
411:   /**
412:    * changes parent of objects
413:    */
414:   void GyrosDebugger::GyrosDebuggerInit(lv_obj_t *parent)
415:   {
416:     //sets parent of container to pointer of new parent
417:     //this is to allow seperation of tabs into seperate classes
418:     //reduce the quantity in one class and to allow for ease of adding
419:     //new or different tabs
420:
421:     lv_obj_set_parent(container, parent);
422:   }
423:
424:
425:   /**
426:    * updates data and shows raw and corrected
427:    */
428:   void GyrosDebugger::update_gyro_info()
429:   {
430:     std::string names_title = "Gyro";
431:     std::string raw_title = "Raw Input";
432:     std::string corrected_title = "Corrected Input";
433:
434:     std::string names = "Chassis Gyro 1\nChassis Gyro 2";
435:
436:     gyroValues raw_vals = getRawGyro();
437:     gyroValues corrected_vals = getCorrectedGyro();
438:
439:     std::string raw = (
440:       std::to_string(raw_vals.gyro1) + "\n" +
441:       std::to_string(raw_vals.gyro2)
442:     );
443:
444:     std::string corrected = (
445:       std::to_string(corrected_vals.gyro1) + "\n" +
446:       std::to_string(corrected_vals.gyro2)
447:     );
448:
449:     lv_label_set_text(title1, names_title.c_str());
450:     lv_label_set_text(title2, raw_title.c_str());
451:     lv_label_set_text(title3, corrected_title.c_str());
452:     lv_label_set_text(info1, names.c_str());
```

```
453:     lv_label_set_text(info2, raw.c_str());
454:     lv_label_set_text(info3, corrected.c_str());
455: }
456:
457:
458:
459:
460:
461:
462: PotentiometerDebugger::PotentiometerDebugger()
463: {
464: //init container
465:     container = lv_cont_create(lv_scr_act(), NULL);
466:     lv_cont_set_fit(container, false, false);
467:     lv_obj_set_style(container, &gray);
468:     lv_cont_set_fit(container, false, false);
469:     lv_obj_set_width(container, SENSORS_CONTAINER_WIDTH);
470:     lv_obj_set_height(container, SENSORS_CONTAINER_HEIGHT);
471:
472: //title for columns
473:     //1
474:     title1 = lv_label_create(container, NULL);
475:     lv_obj_set_style(title1, &toggle_tabbtn_pressed);
476:     lv_obj_set_width(title1, (SENSORS_CONTAINER_WIDTH));
477:     lv_obj_set_height(title1, 20);
478:     lv_label_set_align(title1, LV_LABEL_ALIGN_CENTER);
479:     lv_label_set_text(title1, "None");
480:
481:     //2
482:     title2 = lv_label_create(container, NULL);
483:     lv_obj_set_style(title2, &toggle_tabbtn_pressed);
484:     lv_obj_set_width(title2, (SENSORS_CONTAINER_WIDTH/3));
485:     lv_obj_set_height(title2, 20);
486:     lv_label_set_align(title2, LV_LABEL_ALIGN_CENTER);
487:     lv_label_set_text(title2, "None");
488:
489:     //3
490:     title3 = lv_label_create(container, NULL);
491:     lv_obj_set_style(title3, &toggle_tabbtn_pressed);
492:     lv_obj_set_width(title3, (SENSORS_CONTAINER_WIDTH/3));
493:     lv_obj_set_height(title3, 20);
494:     lv_label_set_align(title3, LV_LABEL_ALIGN_CENTER);
495:     lv_label_set_text(title3, "None");
496:
497: //info for columns
498:     //1
499:     info1 = lv_label_create(container, NULL);
500:     lv_obj_set_style(info1, &toggle_tabbtn_pressed);
501:     lv_obj_set_width(info1, (SENSORS_CONTAINER_WIDTH/3));
502:     lv_obj_set_height(info1, SENSORS_CONTAINER_HEIGHT - 20);
503:     lv_label_set_align(info1, LV_LABEL_ALIGN_LEFT);
504:     lv_label_set_text(info1, "None");
505:
506:     //2
507:     info2 = lv_label_create(container, NULL);
508:     lv_obj_set_style(info2, &toggle_tabbtn_pressed);
509:     lv_obj_set_width(info2, (SENSORS_CONTAINER_WIDTH / 3));
510:     lv_obj_set_height(info2, SENSORS_CONTAINER_HEIGHT - 20);
511:     lv_label_set_align(info2, LV_LABEL_ALIGN_LEFT);
512:     lv_label_set_text(info2, "None");
513:
514:     //3
515:     info3 = lv_label_create(container, NULL);
516:     lv_obj_set_style(info3, &toggle_tabbtn_pressed);
517:     lv_obj_set_width(info3, (SENSORS_CONTAINER_WIDTH/3));
518:     lv_obj_set_height(info3, SENSORS_CONTAINER_HEIGHT - 20);
519:     lv_label_set_align(info3, LV_LABEL_ALIGN_LEFT);
520:     lv_label_set_text(info3, "None");
521:
522: //calibrate button
523:     //button
524:     btn_calibrate = lv_btn_create(container, NULL);
525:     lv_btn_set_style(btn_calibrate, LV_BTN_STYLE_REL, &toggle_btn_released);
526:     lv_btn_set_style(btn_calibrate, LV_BTN_STYLE_PR, &toggle_btn_pressed);
527:     lv_btn_set_action(btn_calibrate, LV_BTN_ACTION_CLICK, btn_calibrate_action);
528:     lv_obj_set_width(btn_calibrate, 110);
529:     lv_obj_set_height(btn_calibrate, 25);
530:
531:     //label
532:     btn_calibrate_label = lv_label_create(btn_calibrate, NULL);
533:     lv_obj_set_style(btn_calibrate_label, &subheading_text);
534:     lv_label_set_text(btn_calibrate_label, "Calibrate");
535:
536: //set positions relative to container
537:     lv_obj_align(title1, container, LV_ALIGN_IN_TOP_LEFT, 10, 10);
538:     lv_obj_align(info1, container, LV_ALIGN_IN_TOP_LEFT, 10, 30);
539:
540:     lv_obj_align(title2, container, LV_ALIGN_IN_TOP_MID, -15, 10);
541:     lv_obj_align(info2, container, LV_ALIGN_IN_TOP_MID, -15, 30);
542:
543:     lv_obj_align(title3, container, LV_ALIGN_IN_TOP_RIGHT, -100, 10);
544:     lv_obj_align(info3, container, LV_ALIGN_IN_TOP_RIGHT, -100, 30);
545:
546:     lv_obj_align(btn_calibrate, container, LV_ALIGN_IN_BOTTOM_RIGHT, -50, 0);
547: }
548:
549: PotentiometerDebugger::~PotentiometerDebugger()
550: {
551:
552: }
553:
554:
555: /**
556:  * calibrates potentiometer and adds loading bar show gui doesn't appear to hang
557:  */
558: lv_res_t PotentiometerDebugger::btn_calibrate_action(lv_obj_t *btn)
559: {
560:     Loading load;
561:     load.show_load(500, lv_scr_act(), 190, 240); //shows loading bar while calibrating
562:     Sensors::calibratePot();
563:     load.hide_load();
564:
565:     return LV_RES_OK;
```

```
566:  }
567:
568:
569:  /**
570:   * changes parent of objects
571:   */
572:  void PotentiometerDebugger::PotentiometerDebuggerInit(lv_obj_t *parent)
573:  {
574:      //sets parent of container to pointer of new parent
575:      //this is to allow seperation of tabs into seperate classes
576:      //reduce the quantity in one class and to allow for ease of adding
577:      //new or different tabs
578:
579:      lv_obj_set_parent(container, parent);
580:  }
581:
582:
583:  /**
584:   * updates potentiometer data with raw and corrected values
585:   */
586:  void PotentiometerDebugger::update_pot_info()
587:  {
588:      std::string names_title = "Potentiometer";
589:      std::string raw_title = "Raw Input";
590:      std::string corrected_title = "Corrected Input";
591:
592:      std::string names = "Lift Potentiometer";
593:
594:      std::string raw = (
595:          std::to_string(getRawPot())
596:      );
597:
598:      std::string corrected = (
599:          std::to_string(getCorrectedPot())
600:      );
601:
602:      lv_label_set_text(title1, names_title.c_str());
603:      lv_label_set_text(title2, raw_title.c_str());
604:      lv_label_set_text(title3, corrected_title.c_str());
605:      lv_label_set_text(info1, names.c_str());
606:      lv_label_set_text(info2, raw.c_str());
607:      lv_label_set_text(info3, corrected.c_str());
608:  }
609:
610:
611:
612:
613:  LimitSwitchDebugger::LimitSwitchDebugger()
614:  {
615:  //init container
616:      container = lv_cont_create(lv_scr_act(), NULL);
617:      lv_cont_set_fit(container, false, false);
618:      lv_obj_set_style(container, &gray);
619:      lv_cont_set_fit(container, false, false);
620:      lv_obj_set_width(container, SENSORS_CONTAINER_WIDTH);
621:      lv_obj_set_height(container, SENSORS_CONTAINER_HEIGHT);
622:
623:  //title for columns
624:      //1
625:      title1 = lv_label_create(container, NULL);
626:      lv_obj_set_style(title1, &toggle_tabbtn_pressed);
627:      lv_obj_set_width(title1, (SENSORS_CONTAINER_WIDTH));
628:      lv_obj_set_height(title1, 20);
629:      lv_label_set_align(title1, LV_LABEL_ALIGN_CENTER);
630:      lv_label_set_text(title1, "None");
631:
632:      //2
633:      title2 = lv_label_create(container, NULL);
634:      lv_obj_set_style(title2, &toggle_tabbtn_pressed);
635:      lv_obj_set_width(title2, (SENSORS_CONTAINER_WIDTH/3));
636:      lv_obj_set_height(title2, 20);
637:      lv_label_set_align(title2, LV_LABEL_ALIGN_CENTER);
638:      lv_label_set_text(title2, "None");
639:
640:  //info for columns
641:      //1
642:      info1 = lv_label_create(container, NULL);
643:      lv_obj_set_style(info1, &toggle_tabbtn_pressed);
644:      lv_obj_set_width(info1, (SENSORS_CONTAINER_WIDTH/3));
645:      lv_obj_set_height(info1, SENSORS_CONTAINER_HEIGHT - 20);
646:      lv_label_set_align(info1, LV_LABEL_ALIGN_LEFT);
647:      lv_label_set_text(info1, "None");
648:
649:      //2
650:      info2 = lv_label_create(container, NULL);
651:      lv_obj_set_style(info2, &toggle_tabbtn_pressed);
652:      lv_obj_set_width(info2, (SENSORS_CONTAINER_WIDTH/3));
653:      lv_obj_set_height(info2, SENSORS_CONTAINER_HEIGHT - 20);
654:      lv_label_set_align(info2, LV_LABEL_ALIGN_LEFT);
655:      lv_label_set_text(info2, "None");
656:
657:
658:  //set positions relative to container
659:      lv_obj_align(title1, container, LV_ALIGN_IN_TOP_LEFT, 10, 10);
660:      lv_obj_align(info1, container, LV_ALIGN_IN_TOP_LEFT, 10, 30);
661:
662:      lv_obj_align(title2, container, LV_ALIGN_IN_TOP_RIGHT, -100, 10);
663:      lv_obj_align(info2, container, LV_ALIGN_IN_TOP_RIGHT, -100, 30);
664:
665:  }
666:
667:  LimitSwitchDebugger::~LimitSwitchDebugger()
668:  {
669:
670:  }
671:
672:
673:  /**
674:   * changes parent of objects
675:   */
676:  void LimitSwitchDebugger::LimitSwitchDebuggerInit(lv_obj_t *parent)
677:  {
678:      //sets parent of container to pointer of new parent
```

```
679:     //this is to allow seperation of tabs into seperate classes
680:     //reduce the quantity in one class and to allow for ease of adding
681:     //new or different tabs
682:
683:     lv_obj_set_parent(container, parent);
684: }
685:
686:
687: /**
688:  * shows value of limit switch as either 0 or 1
689:  */
690: void LimitSwitchDebugger::update_limit_switch_info()
691: {
692:     std::string names_title = "Limit Switch";
693:     std::string val_title = "State";
694:
695:     std::string names = "Limit Switch 1";
696:
697:     std::string val = (
698:         std::to_string(getLimitSwitch())
699:     );
700:
701:
702:     lv_label_set_text(title1, names_title.c_str());
703:     lv_label_set_text(title2, val_title.c_str());
704:     lv_label_set_text(info1, names.c_str());
705:     lv_label_set_text(info2, val.c_str());
706: }
707:
708:
709:
710:
711: LEDDebugger::LEDDebugger()
712: {
713: //init container
714:     container = lv_cont_create(lv_scr_act(), NULL);
715:     lv_cont_set_fit(container, false, false);
716:     lv_obj_set_style(container, &gray);
717:     lv_cont_set_fit(container, false, false);
718:     lv_obj_set_width(container, SENSORS_CONTAINER_WIDTH);
719:     lv_obj_set_height(container, SENSORS_CONTAINER_HEIGHT);
720:
721: //set button
722:     //button
723:     btn_set = lv_btn_create(container, NULL);
724:     lv_btn_set_style(btn_set, LV_BTN_STYLE_REL, &toggle_btn_released);
725:     lv_btn_set_style(btn_set, LV_BTN_STYLE_PR, &toggle_btn_pressed);
726:     lv_btn_set_action(btn_set, LV_BTN_ACTION_CLICK, btn_set_action);
727:     lv_obj_set_width(btn_set, 160);
728:     lv_obj_set_height(btn_set, 40);
729:
730:     //label
731:     btn_set_label = lv_label_create(btn_set, NULL);
732:     lv_obj_set_style(btn_set_label, &subheading_text);
733:     lv_label_set_text(btn_set_label, "Set");
734:
735: //clear button
736:     //button
737:     btn_clear = lv_btn_create(container, NULL);
738:     lv_btn_set_style(btn_clear, LV_BTN_STYLE_REL, &toggle_btn_released);
739:     lv_btn_set_style(btn_clear, LV_BTN_STYLE_PR, &toggle_btn_pressed);
740:     lv_btn_set_action(btn_clear, LV_BTN_ACTION_CLICK, btn_clear_action);
741:     lv_obj_set_width(btn_clear, 160);
742:     lv_obj_set_height(btn_clear, 40);
743:
744:     //label
745:     btn_clear_label = lv_label_create(btn_clear, NULL);
746:     lv_obj_set_style(btn_clear_label, &subheading_text);
747:     lv_label_set_text(btn_clear_label, "Clear");
748:
749: //set relative positions
750:     lv_obj_align(btn_set, container, LV_ALIGN_IN_TOP_LEFT, 40, 20);
751:     lv_obj_align(btn_clear, container, LV_ALIGN_IN_TOP_RIGHT, -40, 20);
752:
753:
754: }
755:
756: LEDDebugger::~LEDDebugger()
757: {
758:
759: }
760:
761:
762: /**
763:  * changes state of LED to on
764:  */
765: lv_res_t LEDDebugger::btn_set_action(lv_obj_t *btn)
766: {
767:     led.set_value(true);
768:     led_status = true;
769:     return LV_RES_OK;
770: }
771:
772:
773: /**
774:  * changes state of LED to off
775:  */
776: lv_res_t LEDDebugger::btn_clear_action(lv_obj_t *btn)
777: {
778:     led.set_value(false);
779:     led_status = false;
780:     return LV_RES_OK;
781: }
782:
783:
784: /**
785:  * changes parent of objects
786:  */
787: void LEDDebugger::LEDDebuggerInit(lv_obj_t *parent)
788: {
789:     //sets parent of container to pointer of new parent
790:     //this is to allow seperation of tabs into seperate classes
791:     //reduce the quantity in one class and to allow for ease of adding
```

```
792:     //new or different tabs
793:
794:     lv_obj_set_parent(container, parent);
795:   }
796:
797:
798:
799:
800:
801:   VisionSensorDebugger::VisionSensorDebugger()
802:   {
803:   //init screen
804:     vision_sensor_screen = lv_obj_create(NULL, NULL);
805:     lv_obj_set_style(vision_sensor_screen, &gray);
806:
807:   //init title label
808:     title_label = lv_label_create(vision_sensor_screen, NULL);
809:     lv_label_set_style(title_label, &heading_text);
810:     lv_obj_set_width(title_label, SENSORS_CONTAINER_WIDTH);
811:     lv_obj_set_height(title_label, 20);
812:     lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
813:     lv_label_set_text(title_label, "Vision Sensor - Debug");
814:
815:   //init back button
816:     //button
817:     btn_back = lv_btn_create(vision_sensor_screen, NULL);
818:     lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
819:     lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
820:     lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
821:     lv_obj_set_width(btn_back, 75);
822:     lv_obj_set_height(btn_back, 25);
823:
824:     //label
825:     btn_back_label = lv_label_create(btn_back, NULL);
826:     lv_obj_set_style(btn_back_label, &heading_text);
827:     lv_label_set_text(btn_back_label, "Back");
828:
829:   //set positions
830:     lv_obj_set_pos(btn_back, 30, 210);
831:
832:     lv_obj_set_pos(title_label, 180, 5);
833:   }
834:
835:   VisionSensorDebugger::~VisionSensorDebugger()
836:   {
837:     lv_obj_del(vision_sensor_screen);
838:   }
839:
840:   lv_res_t VisionSensorDebugger::btn_back_action(lv_obj_t *btn)
841:   {
842:     cont = false;
843:     return LV_RES_OK;
844:   }
845:
846:
847:   /**
848:    * loads page for sensor and waits for user to hit the back button for
849:    * loop to break
850:    */
851:   void VisionSensorDebugger::load_vision_sensor_page()
852:   {
853:     cont = true;
854:
855:     lv_scr_load(vision_sensor_screen);
856:
857:     while ( cont )
858:     {
859:       pros::delay(200);
860:     }
861:
862:   }
863:
864:
865:
866:
867:   SensorsDebug::SensorsDebug()
868:   {
869:   //set default for statics
870:     all_cont = true;
871:
872:   //init screen
873:     sensors_debug_screen = lv_obj_create(NULL, NULL);
874:     lv_obj_set_style(sensors_debug_screen, &gray);
875:
876:   //init title label
877:     title_label = lv_label_create(sensors_debug_screen, NULL);
878:     lv_label_set_style(title_label, &heading_text);
879:     lv_obj_set_width(title_label, SENSORS_CONTAINER_WIDTH);
880:     lv_obj_set_height(title_label, 20);
881:     lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
882:     lv_label_set_text(title_label, "Sensors - Debug");
883:
884:   //init tabview
885:     tabview = lv_tabview_create(sensors_debug_screen, NULL);
886:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BG, &gray);
887:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BTN_REL, &toggle_tabbtn_released);
888:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BTN_PR, &toggle_tabbtn_pressed);
889:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_INDIC, &sw_indic);
890:     lv_tabview_set_style(tabview, LV_TABVIEW_STYLE_BTN_TGL_REL, &toggle_tabbtn_pressed);
891:     //lv_tabview_set_tab_load_action(tabview, tab_load_action);
892:     lv_obj_set_width(tabview, SENSORS_CONTAINER_WIDTH);
893:     lv_obj_set_height(tabview, 200);
894:
895:   //init tabs
896:     imes_tab = lv_tabview_add_tab(tabview, "Encoders");
897:     acclr_tab = lv_tabview_add_tab(tabview, "Accel");
898:     gyros_tab = lv_tabview_add_tab(tabview, "Gyros");
899:     pot_tab = lv_tabview_add_tab(tabview, "Pot");
900:     limit_tab = lv_tabview_add_tab(tabview, "Limit\nSwitch");
901:     led_tab = lv_tabview_add_tab(tabview, "LED");
902:     vision_sensor_tab = lv_tabview_add_tab(tabview, "Vision\nSensor");
903:
904:
```

```
905:  //init back button
906:     //button
907:     btn_back = lv_btn_create(sensors_debug_screen, NULL);
908:     lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
909:     lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
910:     lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
911:     lv_obj_set_width(btn_back, 75);
912:     lv_obj_set_height(btn_back, 25);
913:
914:     //label
915:     btn_back_label = lv_label_create(btn_back, NULL);
916:     lv_obj_set_style(btn_back_label, &heading_text);
917:     lv_label_set_text(btn_back_label, "Back");
918:
919:  //init tabs from other classes
920:     IMEsDebuggerInit(imes_tab);
921:     AccelerometerDebuggerInit(acclr_tab);
922:     GyrosDebuggerInit(gyros_tab);
923:     PotentiometerDebuggerInit(pot_tab);
924:     LimitSwitchDebuggerInit(limit_tab);
925:     LEDDebuggerInit(led_tab);
926:
927:  //set positions
928:     lv_obj_set_pos(btn_back, 30, 210);
929:
930:     lv_obj_set_pos(title_label, 180, 5);
931:
932:     lv_obj_set_pos(tabview, 20, 25);
933:  }
934:
935:
936:  SensorsDebug::~SensorsDebug()
937:  {
938:     //deletes widgets instantiated by class
939:     lv_obj_del(title_label);
940:
941:     lv_obj_del(btn_back_label);
942:     lv_obj_del(btn_back);
943:
944:     lv_obj_del(imes_tab);
945:     lv_obj_del(acclr_tab);
946:     lv_obj_del(gyros_tab);
947:     lv_obj_del(pot_tab);
948:     lv_obj_del(limit_tab);
949:     lv_obj_del(led_tab);
950:     lv_obj_del(vision_sensor_tab);
951:
952:     lv_obj_del(tabview);
953:
954:     lv_obj_del(sensors_debug_screen);
955:  }
956:
957:
958:  /**
959:   * callback funciton that exits main loop when button is pressed
960:   */
961:  lv_res_t SensorsDebug::btn_back_action(lv_obj_t *btn)
962:  {
963:     all_cont = 0;
964:     return LV_RES_OK;
965:  }
966:
967:
968:
969:  /**
970:   * switches on tab loaded, this corresponds to a sensor tab
971:   * if this sensor needs to be calibrated then there is a warning box that
972:   * lets the user choosed to calibrate the sensor, and will not allow the user
973:   * to access the tab until the sensor is calibrated
974:   */
975:  void SensorsDebug::debug()
976:  {
977:     //used to check if user wants to continue cycling through
978:     //tabs. Will be set to zero and loop will break if user hits
979:     //the back button
980:     all_cont = 1;
981:
982:     lv_tabview_set_tab_act(tabview, 0, NULL);
983:     lv_scr_load(sensors_debug_screen);
984:
985:     while ( all_cont )
986:     {
987:        switch ( lv_tabview_get_tab_act(tabview) ) //switches to tab user wants to go to
988:        {
989:           case 0:
990:              update_imes_info();
991:              break;
992:
993:           case 1:
994:              if ( !(accelCalibrated) ) //checks for sensor being
995:                                        //calibrated. If not warning
996:                                        //will appear
997:              {
998:                 lv_tabview_set_sliding(tabview, false); //dissallows changing
999:                                                         //tab until user
1000:                                                        //has selected a
1001:                                                        //calibrate option
1002:
1003:                 std::string msg = (
1004:                    "Accelerometer has not been calibrated.\n"
1005:                    "Click continue to calibrate, or back to\n"
1006:                    "return to a previous screen\n\n"
1007:                    "(Please keep sensor still while calibrating)\n"
1008:                 );
1009:
1010:                 WarningMessage warnmsg;
1011:                 bool calibrated = warnmsg.warn(msg, sensors_debug_screen);
1012:
1013:                 lv_tabview_set_sliding(tabview, true); //re-enables switching
1014:                                                        //tabs
1015:
1016:                 if ( calibrated )
1017:                 {
```

```
1018:                 Loading load;
1019:                 load.show_load(1800, sensors_debug_screen, 190, 125); //shows loading circle while calibrating
1020:                 Sensors::calibrateAccel();
1021:                 load.hide_load();
1022:
1023:                 update_accelerometer_info();
1024:               }
1025:
1026:               else
1027:               {
1028:                 lv_tabview_set_tab_act(tabview, 0, NULL);
1029:                 //tab_loaded = 0;
1030:               }
1031:             }
1032:
1033:             else //if Accelerometer is already calibrated
1034:             {
1035:               update_accelerometer_info();
1036:             }
1037:
1038:             break;
1039:
1040:
1041:           case 2:
1042:             if ( !(gyroCalibrated) )  //checks for sensor being
1043:                                       //calibrated. If not warning
1044:                                       //will appear
1045:             {
1046:               lv_tabview_set_sliding(tabview, false); //dissallows changing
1047:                                                       //tab until user
1048:                                                       //has selected a
1049:                                                       //calibrate option
1050:
1051:               std::string msg = (
1052:                 "Gyros have not been calibrated.\n"
1053:                 "Click continue to calibrate, or back to\n"
1054:                 "return to a previous screen\n\n"
1055:                 "(Please keep sensor still while calibrating)\n"
1056:               );
1057:
1058:               WarningMessage warnmsg;
1059:               bool calibrated = warnmsg.warn(msg, sensors_debug_screen);
1060:
1061:               lv_tabview_set_sliding(tabview, true); //re-enables switching
1062:                                                      //tabs
1063:
1064:               if ( calibrated )
1065:               {
1066:                 Loading load;
1067:                 load.show_load(1000, sensors_debug_screen, 190, 125); //shows loading bar while calibrating
1068:                 Sensors::calibrateGyro();
1069:                 load.hide_load();
1070:
1071:                 update_gyro_info();
1072:               }
1073:
1074:               else
1075:               {
1076:                 lv_tabview_set_tab_act(tabview, 0, NULL);
1077:                 //tab_loaded = 0;
1078:               }
1079:             }
1080:
1081:             else //if Accelerometer is already calibrated
1082:             {
1083:               update_gyro_info();
1084:             }
1085:
1086:             break;
1087:
1088:           case 3:
1089:             if ( !(potCalibrated) )   //checks for sensor being
1090:                                       //calibrated. If not warning
1091:                                       //will appear
1092:             {
1093:               lv_tabview_set_sliding(tabview, false); //dissallows changing
1094:                                                       //tab until user
1095:                                                       //has selected a
1096:                                                       //calibrate option
1097:
1098:               std::string msg = (
1099:                 "Potentiometer has not been calibrated.\n"
1100:                 "Click continue to calibrate, or back to\n"
1101:                 "return to a previous screen\n\n"
1102:                 "(Please keep sensor still while calibrating)\n"
1103:               );
1104:
1105:               WarningMessage warnmsg;
1106:               bool calibrated = warnmsg.warn(msg, sensors_debug_screen);
1107:
1108:               lv_tabview_set_sliding(tabview, true); //re-enables switching
1109:                                                      //tabs
1110:
1111:               if ( calibrated )
1112:               {
1113:                 Loading load;
1114:                 load.show_load(500, sensors_debug_screen, 190, 125); //shows loading circle while calibrating
1115:                 Sensors::calibratePot();
1116:                 load.hide_load();
1117:
1118:                 update_pot_info();
1119:               }
1120:
1121:               else
1122:               {
1123:                 lv_tabview_set_tab_act(tabview, 0, NULL);
1124:                 //tab_loaded = 0;
1125:               }
1126:             }
1127:
1128:             else //if Accelerometer is already calibrated
1129:             {
1130:               update_pot_info();
```

```
1131:                 }
1132:
1133:                 break;
1134:
1135:             case 4:
1136:                 update_limit_switch_info();
1137:                 break;
1138:             case 5:
1139:                 //update led info
1140:                 //no function needed because it is self contained in the led
1141:                 //class
1142:                 break;
1143:             case 6:
1144:                 load_vision_sensor_page();
1145:                 lv_scr_load(sensors_debug_screen);
1146:
1147:                 //switch to a different tab or user will be unable to leave
1148:                 //vision sensor debugger
1149:                 lv_tabview_set_tab_act(tabview, 0, NULL);
1150:                 //tab_loaded = 0;
1151:                 break;
1152:         }
1153:
1154:         pros::delay(200);
1155:     }
1156: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/TitleScreen.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2015
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * contains class that allows user to select a debug tab
8:   *
9:   */
10:
11: #ifndef __TITLESCREEN_HPP__
12: #define __TITLESCREEN_HPP__
13:
14:
15: #include "../../../../include/main.h"
16:
17: #include "../Styles.hpp"
18:
19:
20: /**
21:  * @see: ../Styles.hpp
22:  * @see: Debug.hpp
23:  *
24:  * contains button matrix that has different debug tabs on it
25:  */
26: class TitleScreen : private Styles
27: {
28:     private:
29:         //screen
30:         lv_obj_t *title_screen;
31:
32:         lv_obj_t *title_label;
33:
34:         lv_obj_t *btn_back;
35:         lv_obj_t *btn_back_label;
36:
37:         /**
38:          * @param: lv_obj_t* btn -> button that called the funtion
39:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
40:          *
41:          * button callback function used to set the debug option to -1 meaning the
42:          * user wants to go to the previous screen
43:          */
44:         static lv_res_t btn_back_action(lv_obj_t *btn);
45:
46:
47:         lv_obj_t *button_matrix; //button matrix object
48:         static const char* btnm_map[]; //map for button matrix
49:
50:         /**
51:          * @param: lv_obj_t* btn -> button that called the funtion
52:          * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
53:          *
54:          * button callback function used to set option of debug screen that user wants to go to
55:          */
56:         static lv_res_t button_matrix_action(lv_obj_t *btnm, const char *btn_txt);
57:
58:
59:     public:
60:         TitleScreen();
61:         ~TitleScreen();
62:
63:         static int option;
64:
65:         /**
66:          * @return: None
67:          *
68:          * @see: btn_back_action
69:          * @see: button_matrix_action
70:          *
71:          * loads screen and waits in a loop with a delay for user to select
72:          * a button
73:          */
74:         void chooseOption();
75:
76: };
77:
78: #endif
```

```
/**
 * @file: ./RobotCode/src/lcdCode/Debug/TitleScreen.cpp
 * @author: Aiden Carney
 * @reviewed_on: 10/15/2019
 * @reviewed_by: Aiden Carney
 *
 * @see: TitleScreen.hpp
 *
 * contains class for selecting a debug screen or going to previous stage
 */

#include "../../../../include/main.h"
#include "../../../../include/api.h"

#include "TitleScreen.hpp"
#include "../../controller/controller.hpp"


int TitleScreen::option = 0;
const char* TitleScreen::btnm_map[] = {
      "Motors", "Sensors", "Controller", "Battery",
      "\n", "Field Control", "Wiring", "Internal\nMotor PID", ""
      };

TitleScreen::TitleScreen()
{
   option = 0;

   title_screen = lv_obj_create(NULL, NULL);
   lv_obj_set_style(title_screen, &gray);

//init button matrix
   button_matrix = lv_btnm_create(title_screen, NULL);
   lv_btnm_set_map(button_matrix, btnm_map);
   lv_btnm_set_action(button_matrix, button_matrix_action);
   lv_obj_set_width(button_matrix, 440);
   lv_obj_set_height(button_matrix, 140);

   //set styles of button matrix
   lv_btnm_set_style(button_matrix, LV_BTNM_STYLE_BTN_REL, &toggle_btn_released);
   lv_btnm_set_style(button_matrix, LV_BTNM_STYLE_BTN_PR, &toggle_btn_pressed);

//init title label
   title_label = lv_label_create(title_screen, NULL);
   lv_obj_set_style(title_label, &heading_text);
   lv_obj_set_width(title_label, 300);
   lv_obj_set_height(title_label, 20);
   lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
   lv_label_set_text(title_label, "Debugger");

//init back button
   //button
   btn_back = lv_btn_create(title_screen, NULL);
   lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
   lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
   lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
   lv_obj_set_width(btn_back, 75);
   lv_obj_set_height(btn_back, 25);

   //label
   btn_back_label = lv_label_create(btn_back, NULL);
   lv_obj_set_style(btn_back_label, &heading_text);
   lv_label_set_text(btn_back_label, "Back");


//set postitions of widgets
   lv_obj_set_pos(btn_back, 210, 200);
   lv_obj_set_pos(title_label, 210, 20);
   lv_obj_set_pos(button_matrix, 20, 50);

}

TitleScreen::~TitleScreen()
{
   lv_obj_del(btn_back_label);
   lv_obj_del(btn_back);
   lv_obj_del(title_label);
   lv_obj_del(button_matrix);

   lv_obj_del(title_screen);
}



/**
 * compares text of button to text of label to see what button was clicked
 * sets int option to value based on the button clicked
 */
lv_res_t TitleScreen::button_matrix_action(lv_obj_t *btnm, const char *btn_txt)
{
   if (btn_txt == "Motors")
   {
      option = 1;
   }
   else if (btn_txt == "Sensors")
   {
      option = 2;
   }
   else if (btn_txt == "Controller")
   {
      option = 3;
   }
   else if (btn_txt == "Battery")
   {
      option = 4;
   }
   else if (btn_txt == "Field Control")
   {
      option = 5;
   }
   else if (btn_txt == "Wiring")
   {
```

```
114:         option = 6;
115:     }
116:     else if (btn_txt == "Internal\nMotor PID")
117:     {
118:         option = 7;
119:     }
120:     return LV_RES_OK;
121: }
122:
123:
124:
125: /**
126:  * sets option to -1 which is to be interpreted as user wanting to go back
127:  */
128: lv_res_t TitleScreen::btn_back_action(lv_obj_t *btn)
129: {
130:     option = -1;
131:     return LV_RES_OK;
132: }
133:
134:
135:
136: /**
137:  * waits for option to be non zero
138:  * this will happen once any button is clicked
139:  */
140: void TitleScreen::chooseOption()
141: {
142:     Controller controllers;
143:     option = 0;
144:
145:     lv_scr_load(title_screen);
146:     while ( !option )
147:     {
148:         //allow controller to press the buttons as well
149:         if ( controllers.master.get_digital(pros::E_CONTROLLER_DIGITAL_B) )
150:         {
151:             btn_back_action( NULL );
152:             pros::delay(100);
153:         }
154:         pros::delay(20);
155:     }
156: }
```

```
1:  /**
2:   * @file: ./RobotCode/src/lcdCode/Debug/Wiring.hpp
3:   * @author: Aiden Carney
4:   * @reviewed_on: 10/15/2019
5:   * @reviewed_by: Aiden Carney
6:   *
7:   * contains class that shows the current wiring of the robot
8:   */
9:
10:  #ifndef __WIRING_HPP__
11:  #define __WIRING_HPP__
12:
13:  #include "../Styles.hpp"
14:
15:  #include "../../motors/Motors.hpp"
16:  #include "../../sensors/Sensors.hpp"
17:
18:
19:  /**
20:   * @see: ../Styles.hpp
21:   * @see: ../../motors/Motors.hpp
22:   * @see: ../../sensors/Sensors.hpp
23:   *
24:   * shows the ports that each motor or sensor is located on
25:   * purpose is to make it easier and more companct to wire the robot than having
26:   * to read off of separate computer screen
27:   */
28:  class Wiring : private Styles
29:  {
30:      private:
31:          lv_obj_t *wiring_screen;
32:          lv_obj_t *title_label;
33:
34:          lv_obj_t *motor_info;
35:          lv_obj_t *sensors_info;
36:
37:          //back button
38:          lv_obj_t *btn_back;
39:          lv_obj_t *btn_back_label;
40:
41:          /**
42:           * @param: lv_obj_t* btn -> button that called the funtion
43:           * @return: lv_res_t -> LV_RES_OK on successfull completion because object still exists
44:           *
45:           * button callback function used to set cont to false meaning the
46:           * user wants to go to the title screen
47:           */
48:          static lv_res_t btn_back_action(lv_obj_t *btn);
49:
50:      public:
51:          static bool cont;
52:
53:          Wiring();
54:          ~Wiring();
55:
56:          /**
57:           * @return: None
58:           *
59:           * passive screen -- loads text and wait for user to go back
60:           */
61:          void debug();
62:
63:  };
64:
65:
66:
67:  #endif
```

```
/**
 * @file: ./RobotCode/src/lcdCode/Debug/Wiring.cpp
 * @author: Aiden Carney
 * @reviewed_on: 10/15/2019
 * @reviewed_by: Aiden Carney
 *
 * @see: Wiring.hpp
 *
 * contains class that shows wiring configuration
 */

#include "../../../../include/main.h"
#include "../../../../include/api.h"

#include "../Styles.hpp"
#include "Wiring.hpp"

#include "../../motors/Motors.hpp"
#include "../../sensors/Sensors.hpp"


bool Wiring::cont = true;

Wiring::Wiring()
{
    Configuration* config = Configuration::get_instance();

    cont = true;

//screen
    wiring_screen = lv_obj_create(NULL, NULL);
    lv_obj_set_style(wiring_screen, &gray);

//init back button
    //button
    btn_back = lv_btn_create(wiring_screen, NULL);
    lv_btn_set_style(btn_back, LV_BTN_STYLE_REL, &toggle_btn_released);
    lv_btn_set_style(btn_back, LV_BTN_STYLE_PR, &toggle_btn_pressed);
    lv_btn_set_action(btn_back, LV_BTN_ACTION_CLICK, btn_back_action);
    lv_obj_set_width(btn_back, 75);
    lv_obj_set_height(btn_back, 25);

    //label
    btn_back_label = lv_label_create(btn_back, NULL);
    lv_obj_set_style(btn_back_label, &heading_text);
    lv_label_set_text(btn_back_label, "Back");

//init title label
    title_label = lv_label_create(wiring_screen, NULL);
    lv_label_set_style(title_label, &heading_text);
    lv_obj_set_width(title_label, 440);
    lv_obj_set_height(title_label, 20);
    lv_label_set_align(title_label, LV_LABEL_ALIGN_CENTER);
    lv_label_set_text(title_label, "Wiring");

//init motor info label
    motor_info = lv_label_create(wiring_screen, NULL);
    lv_label_set_style(motor_info, &subheading_text);
    lv_obj_set_width(motor_info, 220);
    lv_obj_set_height(motor_info, 200);
    lv_label_set_align(motor_info, LV_LABEL_ALIGN_LEFT);

    std::string motors_text = (
          "front right (200 RPM) - " + std::to_string(config->front_right_port) + "\n"
          "back right (200 RPM) - " + std::to_string(config->back_left_port) + "\n"
          "front left  (200 RPM) - " + std::to_string(config->front_left_port) + "\n"
          "back left  (200 RPM) - " + std::to_string(config->back_right_port) + "\n"
          "right intake   (100 RPM) - " + std::to_string(config->left_intake_port) + "\n"
          "left intake    (100 RPM) - " + std::to_string(config->right_intake_port) + "\n"
          "tilter     100 RPM) - " + std::to_string(config->tilter_port) + "\n"
          "lift     (100 RPM) - " + std::to_string(config->lift_port) + "\n"
    );

    lv_label_set_text(motor_info, motors_text.c_str());

//init motor info label
    sensors_info = lv_label_create(wiring_screen, NULL);
    lv_label_set_style(sensors_info, &subheading_text);
    lv_obj_set_width(sensors_info, 220);
    lv_obj_set_height(sensors_info, 200);
    lv_label_set_align(sensors_info, LV_LABEL_ALIGN_LEFT);

    std::string sensors_text = (
       std::string("gyro 1         - ") + GYRO1_PORT + "\n" +
       "gyro 2         - " + GYRO2_PORT + "\n" +
       "accellerometerX - " + ACCELLEROMETERX_PORT + "\n" +
       "accellerometerY - " + ACCELLEROMETERY_PORT + "\n" +
       "accellerometerZ - " + ACCELLEROMETERZ_PORT + "\n" +
       "potentiometer   - " + POTENTIOMETER_PORT + "\n" +
       "limit switch    - " + LIMITSWITCH_PORT + "\n" +
       "LED             - " + LED_PORT + "\n" +
       "vision sensor   - " + std::to_string(VISIONSENSOR_PORT) + "\n"
    );

    lv_label_set_text(sensors_info, sensors_text.c_str());

//set positions
    lv_obj_set_pos(btn_back, 30, 210);

    lv_obj_set_pos(title_label, 220, 5);

    lv_obj_set_pos(motor_info, 20, 25);
    lv_obj_set_pos(sensors_info, 300, 25);


}

Wiring::~Wiring()
{
    lv_obj_del(wiring_screen);
}
```

```
114:  /**
115:   * sets cont to false signifying user wants to go back, main loop will exit
116:   */
117:  lv_res_t Wiring::btn_back_action(lv_obj_t *btn)
118:  {
119:      cont = false;
120:      return LV_RES_OK;
121:  }
122:
123:
124:  /**
125:   * waits for cont to be false which occurs when the user hits the back button
126:   */
127:  void Wiring::debug()
128:  {
129:      cont = true;
130:
131:      lv_scr_load(wiring_screen);
132:
133:      while ( cont )
134:      {
135:          pros::delay(100);
136:      }
137:  }
```