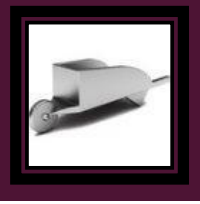
MODULAR MONOPOLY (MOD-OPOLY)

CSCI-3308

Presented By:

- Joel Barkley
- Nathan Carmine
- Ted Freeman
- John Gallagher
- Chase Heck

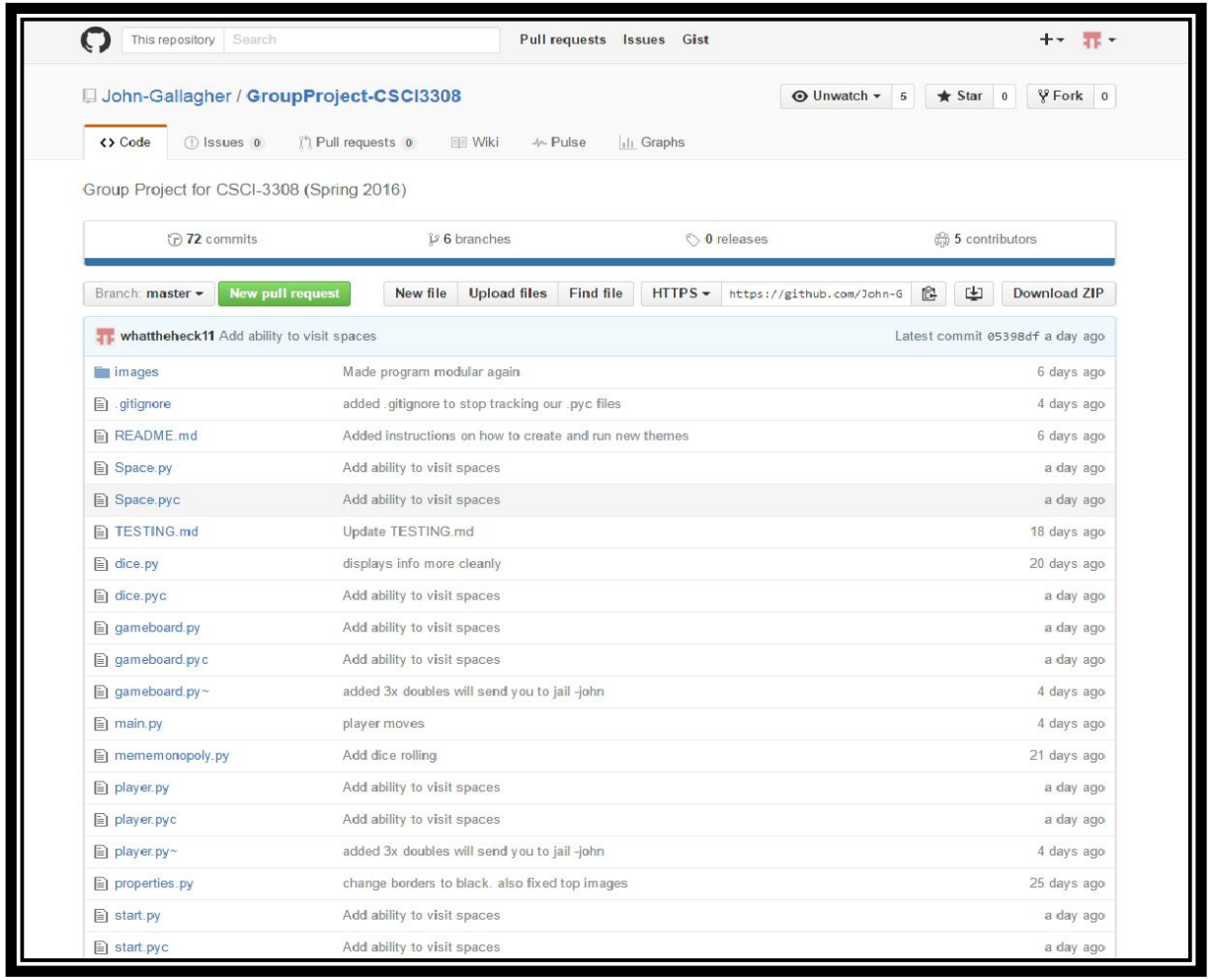





VCS REPOSITORY – GITHUB

- Allowed for tracking of source code files
- All changes and commits by group members are documented, option to reset to previous commits exists in case of errors
- GitHub markdown allows for simple formatting
- Easy version control
- Somewhat confusing terminal commands

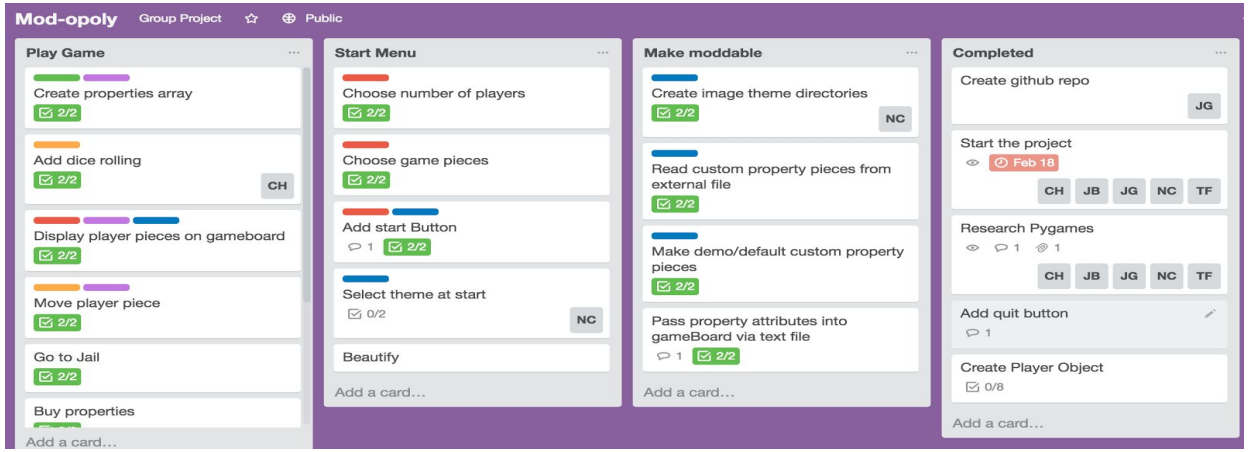
Rating: 4.5 stars

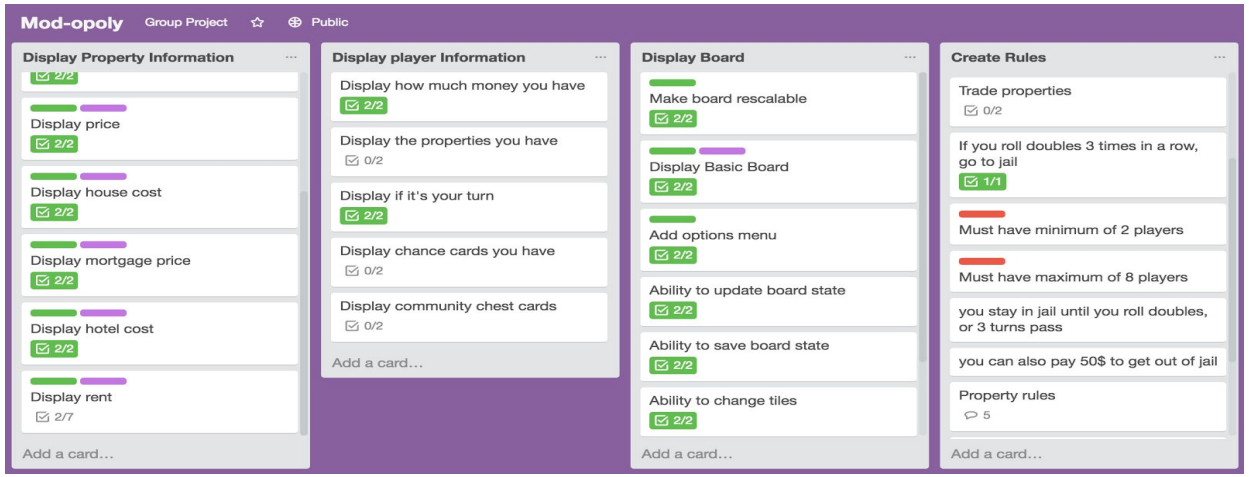


PROJECT TRACKER – TRELLO

- Good for prioritizing tasks in less-rigid fashion
- Very customizable, multiple jobs can be displayed at once according to user stories
- Easy to tell who is working on what and how much of a task is completed
- Easy to set-up and update
- Can become cluttered/too much overhead

Rating: 4 Stars





EXTENDED MODULES – PYGAME

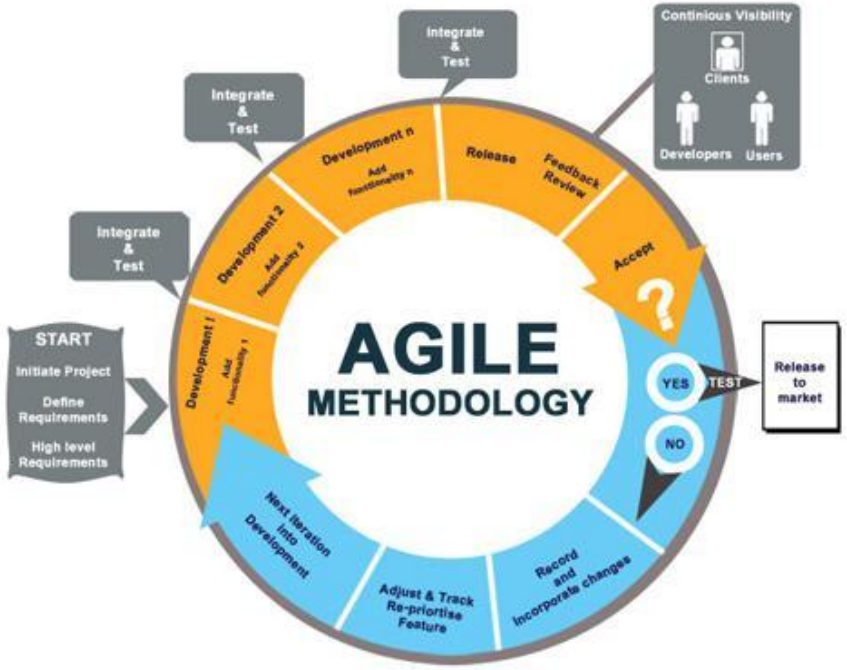
- Much easier game development than basic Python
- Specialized functions for 2D game environments
- Good documentation and tutorials
- 3.5 Stars
- Pros: Basic
- Cons:
  - Certain functionality lacking
  - Anything GUI related
  - Basic

146 lines (127 sloc) | 6.45 KB — Raw | Blame | History

```
import random, sys, pygame, time, copy
from pygame.locals import *
from player import Player
from gameboard import *
from start import *
from sys import argv
import itertools
import os
import csv

#Whenever an error arises, this function is called
def loadSpacesError(isDefault):
    if not isDefault:
        #Tries to load the default theme if an error arises in another theme
        print "Attempting to load default theme"
        return loadSpaces("default")
    else:
        #If there's an issue with loading the default theme, exit the program
        print "Error loading default theme. Please reinstall game files."
        exit(0)

def loadSpaces(theme_name):
    #Lists for each of the six columns in spaces.csv
    space_titles = []
    space_types = []
    space_prices = []
    property_rent = []
    house_prices = []
    property_colors = []
    csv_path = "./images/themes/"+theme_name+"/spaces.csv"
    if os.path.exists(csv_path):
        with open(csv_path, 'rb') as titlesfile:
            #itertools.tee sets up a second reader: One gets the # of cols for each row
            #The other goes through each row and appends spaces to their appropriate lists
            reader1, reader2 = itertools.tee(csv.reader(titlesfile, delimiter=','))
            row_counts = [] #Where the # of cols for each row goes
            purchasable_count = 0 # of purchasable space
            utility_count = 0 # of utility spaces
            property_count = 0 # of property spaces
            for i in range(0,6): #First reader - gets # cols in each row
                row_counts.append(len(reader1.next()))
            del reader1

            #Ensures first two rows have 40 cols - necessary in any monopoly game
            if row_counts[0] < 40 or row_counts[1] < 40:
                print theme_name+"/spaces.csv does not contain titles and/or types for all 40 spaces."
                titlesfile.close()
                return loadSpacesError(theme_name == "default")

            #Begin appending titles and types to their respective lists
            for title in reader2.next():
                space_titles.append(str(title))
```

METHODOLOGY

- Agile User Stories:
  - Tasks weighted by user story rank
  - Independent phases of development according to stories
  - Distinction of functional and non-functional requirements kept development streamlined
- Agile Board:
  - Tasks separated by phase and importance.
  - Multiple tasks could be tackled at once while maintaining project malleability
  - Continuous visibility



CHALLENGES

- Most of us were unfamiliar with Pygame and general game development
- Implementation of more difficult tasks such as trading, player information, etc.
- Unit testing
- Scheduling
- Combining work
- Auto-doc

PROJECT DEMO