CSUDH

DEPARTMENT OF
COMPUTER SCIENCE

Spring 2019

# Senior Project Report

## Department of Computer Science

## California State University, Dominguez Hills

# *Smart Parking*

Prepared By

**Matthew Yamamoto**

In
Partial Fulfillment of the requirements

For
Senior Design – CSC 492

Department of Computer Science
California State University, Dominguez Hills
**Spring 2019**

**Committee Members/Approval**

| | | |
|---|---|---|
| **Amlan Chatterjee** | | |
| ***Faculty Advisor*** | ***Signature*** | ***Date*** |
| | | |
| ***Committee Member*** | ***Signature*** | ***Date*** |
| | | |
| ***Committee Member*** | ***Signature*** | ***Date*** |
| **Mohsen Beheshti** | | |
| ***Department Chair*** | ***Signature*** | ***Date*** |

## **ABSTRACT**

For my senior project I am going to be doing a smart parking system. I will be using OpenCV to do vision tracking and determine whether or not there are parking spots available on campus. I will be using python as my language of choice and use the filters provided from OpenCV. The reason I want to do this project is because finding parking is a big problem, not only for our school but also for any big city or compacted area with limited parking spaces. There are solutions that I have seen in parking structures of malls where they use sensors but this can get very costly and is a lot of effort to install all the modules and upkeep them. I want to be able to solve the parking problem here at Dominguez Hills so that you can easily find an open parking spot and not have to drive up and down every isle to look for a spot.

## ACKNOWLEDGEMENT

I want to thank my family and friends for supporting what my choices and making it easier to finish my project and degree. I also want to thank the professors for teaching me and guiding me through the courses.

# Table of Contents

# 1 Introduction

This purpose of this project is to try and solve the problem of finding open parking spaces. If you are a student at CSUDH, you know that parking can be hard to find when you want to get a close spot. By using computer vision and connecting that to a database and phone app we can find the available parking spots much quicker.

# 2 Background

The parking at California State Dominguez Hills is getting out of hand because there are more and more people coming to the school but the parking lots are not getting any bigger. Students are always struggling for parking and everyone wants to get as close parking to the classrooms as possible. By using computer vision I want to be able to find all the close parking spots so that when students come to park they don't have to go up and down the isles multiple times in hope for an open spot.

## 2.1 Statistics

An article from USA Today in 2017 stated that drivers spent an average of 17 hours a year searching for parking spots which adds up to about $345 wasted in time, gas, and emissions per person.

## 2.2 Applications

This project can be applied to any place with parking spots. The best place would be a single level parking lot that is outside where you can place the camera up high to get the most cars in view. The higher vantage point of the camera, the cheaper the overall cost of the system will be because of less hardware.

## 2.3 Database

The database stores all of the parking spaces that are vacant or occupied. This makes it quick and easy to update all the spots in real time and send that to the app.

# 3 Research

When researching this smart parking topic I mainly came across hardware solutions where there would be sensors on each parking space (usually one for shadow detection to tell if there is something above it and one for metal detection to tell if its a car and not a person or other object). The problem with this solution is that if you have a thousand parking spaces or more then the cost of this system is going to be huge. I could only find 1 or 2 companies that actually used vision for parking space detection but they were very secretive as it is something that would be very useful for many companies/schools.

## 3.1 OpenCV

Getting OpenCV downloaded and running on my Mac is a long and tedious process. I decided to use Python as my language of choice so the things I needed to download are Xcode, Homebrew, Python3, Python Libraries, OpenCV 4.0.1, and the Python Vitrual Environment. I used the OpenCV documentation and tutorials to figure out what type of filters to apply to the video.

## 3.2 Google Firebase Firestore

Firebase Firestore is a scalable NoSQL cloud database to store and sync data for client-side and server-side development. I wanted to store data from all the parking spots and to insert all the spots into the database I used Python. In order to use the data in the app I had to make calls in the app using Dart. The reason I went with Firestore is because it has realtime updates and is scalable.

## 3.3 Android Studio/Flutter/Dart

Flutter is an open-source mobile application development framework and Dart is the language used to program in it. They are both fairly new and the stable version of Flutter (1.0) is less than 6 months old. The reason for choosing Flutter and Dart is that it made making apps for both Android and iOS very simple. Flutter was made to help developers create beautiful cross platform apps quickly.

## 3.4 Parking Detection Algorithm

There are many different ways you can use OpenCV for this project. There is line detection, background deletion, and many other types of filters that can be used for my application. Since I did not know how to use much of anything from the vast library I stuck with a few filters and to save time and resources I did not use any machine learning algorithms to track the cars or find spots autonomously because it would require more advanced hardware than what I have available to have usable frame rates.

# 4 Development

The things I used for this project was OpenCV (open source computer vision library), Python for the computer application, Dart for the mobile app, and Firebase Firestore for the database.

## 4.1 OpenCV

Out of all the filters that OpenCV has available I chose to use Guassian Blur and Greyscale. I wanted to also use background subtraction but because I was using a drone, the background moved around too much and it was hard to find the lines as the software thought the whole image was a background

### 4.1.1 Implementation

In order to find which parking spots were vacant or occupied I averaged the pixels inside a bounding box that I manually put around each parking space to tell if something was inside it. In order to be able to do this I had to find filters first in order to make the video easier to work with. By using Guassian blur which smooths out the images in each frame and turning the images into greyscale, I was then able to take the bounding box of pixels and average those out and since the image is greyscale, if it got darker then that means a car had gone in the box.

### 4.1.2 Challenges

Some of the challenges I had with the solution I used was that I could not determine the difference between a shadow, a person, or a car very well because I had no real object detection and it was only based on pixel color.

## 4.2 Mobile App

I wanted the app to be available to anyone on campus with an Android or iOS device so I made the app in Flutter using the Dart language. This allowed me to create a cross platform app that can be imported into the Google Play Store and the Apple App Store.

### 4.2.1 Implementation

Using Android Studio I was able to creat the Android app using Flutter and Dart. For demonstration purposes and time constraints, I made the app display up to 95 parking spaces from parking lot 7. The app either displays red or green depending on whether or not the parking spot is taken.

### 4.2.2 Challenges

The challenges I faced were not having the resources to make and test the iOS app. Unfortunatley my Macbook laptop storage is very small and I could not export the code to Xcode to make the iOS app because I don't have enough space on the drive. I did all my programming on my windows laptop but it does not support Xcode and making iOS applications. It was also a challenge to find some solutions to implementing the database calls from Firebase Firestore because Dart is such a new language that I did not find a lot of details for it.

## 5 Conclusion & Future Work

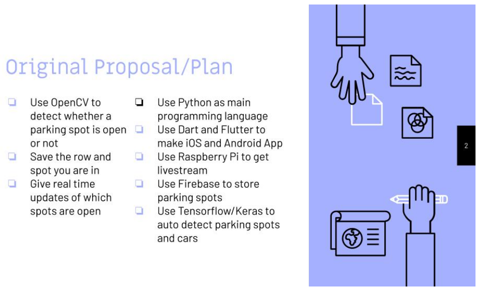
Although my project did what I wanted it to do, it wasn't to the level that I thought it could have been. If I were to redo this project I would want to use machine learning and object detection so I could solve the problem of determine whether there is a car, person, shadow, or other object in the parking spot. I would also want to replace the drone that I used to get the footage for a mounted camera such as a surveilience cctv camera so I could make use of other filters like the background subtraction filter which could help make finding the parking spots easier. This would also make it easier to work with the footage as it won't be swaying and you won't have to try and re-input the parking spots because it will be setup already in the same position as you left it. I would also want to make the iOS app as it should be fairly easy to just export the code from Flutter and Android Studio into Xcode.

# 6 Appendix

## 6.1 Final Presentation

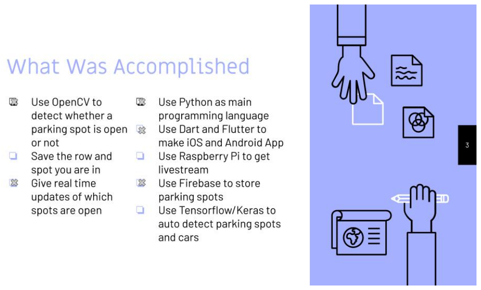
Smart Parking

By Matthew Yamamoto



### Original Proposal/Plan

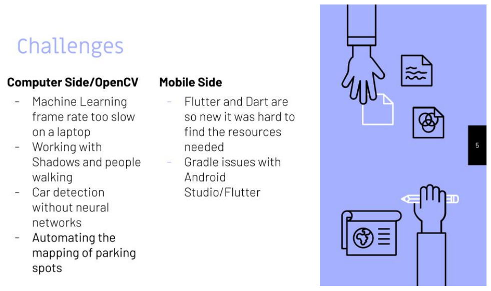
- Use OpenCV to detect whether a parking spot is open or not
- Save the row and spot you are in
- Give real time updates of which spots are open
- Use Python as main programming language
- Use Dart and Flutter to make iOS and Android App
- Use Raspberry Pi to get livestream
- Use Firebase to store parking spots
- Use Tensorflow/Keras to auto detect parking spots and cars



### What Was Accomplished

- Use OpenCV to detect whether a parking spot is open or not
- Save the row and spot you are in
- Give real time updates of which spots are open
- Use Python as main programming language
- Use Dart and Flutter to make iOS and Android App
- Use Raspberry Pi to get livestream
- Use Firebase to store parking spots
- Use Tensorflow/Keras to auto detect parking spots and cars

Challenges Faced



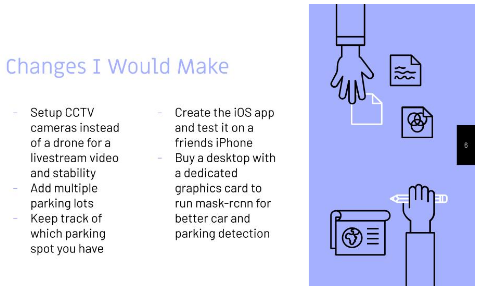
### Challenges

**Computer Side/OpenCV**

- Machine Learning frame rate too slow on a laptop
- Working with Shadows and people walking
- Car detection without neural networks
- Automating the mapping of parking spots

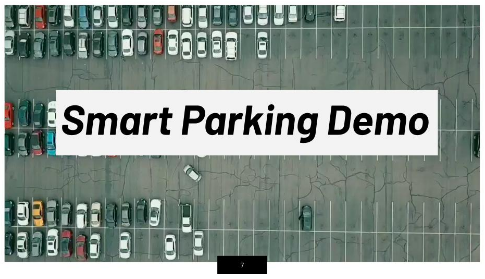
**Mobile Side**

- Flutter and Dart are so new it was hard to find the resources needed
- Gradle issues with Android Studio/Flutter



### Changes I Would Make

- Setup CCTV cameras instead of a drone for a livestream video and stability
- Add multiple parking lots
- Keep track of which parking spot you have
- Create the iOS app and test it on a friends iPhone
- Buy a desktop with a dedicated graphics card to run mask-rcnn for better car and parking detection



Smart Parking Demo

# 7 Code

## 7.1 Python

### 7.1.1 Main

```
import argparse
import yaml
from coordinates_generator import CoordinatesGenerator
from motion_detector import MotionDetector
from colors import *

def main():



    args = parse_args()

    image_file = args.image_file
    data_file = args.data_file
    start_frame = args.start_frame

    if image_file is not None:
    with open(data_file, "w+") as points:
    generator = CoordinatesGenerator(image_file, points, COLOR_RED)
    generator.generate()



    with open(data_file, "r") as data:
    points = yaml.load(data)
    detector = MotionDetector(args.video_file, points,
    int(start_frame))
    detector.detect_motion()



def parse_args():
parser = argparse.ArgumentParser(description='Generates Coordinates
File')

    parser.add_argument("--image",
                        dest="image_file",
                        required=False,
                        help="Image file to generate coordinates on")

    parser.add_argument("--video",
                        dest="video_file",
                        required=True,
                        help="Video file to detect motion on")

    parser.add_argument("--data",
                        dest="data_file",
                        required=True,
                        help="Data file to be used with OpenCV")

    parser.add_argument("--start-frame",
                        dest="start_frame",
                        required=False,
                        default=1,
                        help="Starting frame on the video")
    return parser.parse_args()


if __name__ == '__main__':
    main()
```

### 7.1.2 motion detector

```
import cv2 as open_cv
import numpy as np
import logging
from drawing_utils import draw_contours
from colors import COLOR_GREEN, COLOR_WHITE, COLOR_BLUE
import FirebaseFirestoreTesting as fire


class MotionDetector:

    LAPLACIAN = 1.5
    DETECT_DELAY = 1

    def __init__(self, video, coordinates, start_frame):
        self.video = video
        self.coordinates_data = coordinates
        self.start_frame = start_frame
        self.contours = []
        self.bounds = []
        self.mask = []

    def detect_motion(self):
```

```
lot7A = [bool, bool, bool, bool, bool, bool, bool, bool, bool,
bool, bool, bool, bool, bool, bool, bool, bool, bool, bool,
bool, bool, bool, bool, bool, bool, bool, bool, bool, bool,
bool, bool, bool, bool, bool, bool, bool, bool, bool, bool,
bool, bool, bool, bool, bool, bool, bool, bool, bool, bool,
bool, bool, bool, bool, bool, bool, bool, bool, bool, bool,
bool, bool, bool, bool, bool, bool, bool, bool, bool, bool,
bool, bool, bool, bool, bool, bool, bool, bool, bool, bool,
bool, bool, bool, bool, bool, bool, bool, bool, bool, bool,
bool, bool, bool, bool, bool, bool, bool]
    capture = open_cv.VideoCapture(self.video)
    capture.set(open_cv.CAP_PROP_POS_FRAMES, self.start_frame)

    coordinates_data = self.coordinates_data
    logging.debug("coordinates data: %s", coordinates_data)

    for p in coordinates_data:
        coordinates = self._coordinates(p)
        logging.debug("coordinates: %s", coordinates)

        rect = open_cv.boundingRect(coordinates)
        logging.debug("rect: %s", rect)

        new_coordinates = coordinates.copy()
        new_coordinates[:, 0] = coordinates[:, 0] - rect[0]
        new_coordinates[:, 1] = coordinates[:, 1] - rect[1]
        logging.debug("new_coordinates: %s", new_coordinates)

        self.contours.append(coordinates)
        self.bounds.append(rect)

        mask = open_cv.drawContours(
            np.zeros((rect[3], rect[2]), dtype=np.uint8),
            [new_coordinates],
            contourIdx=-1,
            color=255,
            thickness=-1,
            lineType=open_cv.LINE_8)

        mask = mask == 255
        self.mask.append(mask)
        logging.debug("mask: %s", self.mask)

    statuses = [False] * len(coordinates_data)
    times = [None] * len(coordinates_data)

    while capture.isOpened():
        result, frame = capture.read()
        if frame is None:
            break

        if not result:
        raise CaptureReadError("Error reading video capture on
        frame %s" % str(frame))
        blurred = open_cv.GaussianBlur(frame.copy(), (5, 5), 3)
        grayed = open_cv.cvtColor(blurred, open_cv.COLOR_BGR2GRAY)
        new_frame = frame.copy()
```

```
            logging.debug("new_frame: %s", new_frame)

position_in_seconds = capture.get(open_cv.CAP_PROP_POS_MSEC) /
1000.0

            for index, c in enumerate(coordinates_data):
                status = self.__apply(grayed, index, c)

                if times[index] is not None and
                self.same_status(statuses, index, status):
                    times[index] = None
                    continue

                if times[index] is not None and
                self.status_changed(statuses, index, status):
                    if position_in_seconds - times[index] >=
                    MotionDetector.DETECT_DELAY:
                        statuses[index] = status
                        times[index] = None
                    continue

                if times[index] is None and
                self.status_changed(statuses, index, status):
                    times[index] = position_in_seconds
            for index, p in enumerate(coordinates_data):
                coordinates = self._coordinates(p)

                color = COLOR_GREEN if statuses[index] else
                COLOR_BLUE
                previous = lot7A[index]
                lot7A[index] = True if statuses[index] else False
                if previous != lot7A[index]:
                    fire.update(index, lot7A[index])
                draw_contours(new_frame, coordinates, str(p["id"] +
                1), COLOR_WHITE, color)

            open_cv.imshow(str(self.video), new_frame)
            k = open_cv.waitKey(1)
            if k == ord("q"):
                break
        capture.release()
        open_cv.destroyAllWindows()

    def __apply(self, grayed, index, p):
        coordinates = self._coordinates(p)
        logging.debug("points: %s", coordinates)

        rect = self.bounds[index]
        logging.debug("rect: %s", rect)

        roi_gray = grayed[rect[1]:(rect[1] + rect[3]), rect[0]:
        (rect[0] + rect[2])]
        laplacian = open_cv.Laplacian(roi_gray, open_cv.CV_64F)
        logging.debug("laplacian: %s", laplacian)

        coordinates[:, 0] = coordinates[:, 0] - rect[0]
        coordinates[:, 1] = coordinates[:, 1] - rect[1]
        status = np.mean(np.abs(laplacian * self.mask[index])) <
        MotionDetector.LAPLACIAN
        print(status)
        logging.debug("status: %s", status)

        return status

    @staticmethod
    def _coordinates(p):
        return np.array(p["coordinates"])

    @staticmethod
    def same_status(coordinates_status, index, status):
        return status == coordinates_status[index]

    @staticmethod
    def status_changed(coordinates_status, index, status):
        return status != coordinates_status[index]


class CaptureReadError(Exception):
    pass
```

### 7.1.3 drawing utils

```
import cv2 as open_cv
from colors import COLOR_RED



def draw_contours(image,
                    coordinates,
                    label,
                    font_color,
                    border_color=COLOR_RED,
                    line_thickness=1,
                    font=open_cv.FONT_HERSHEY_SIMPLEX,
                    font_scale=0.5):
    open_cv.drawContours(image,
                         [coordinates],
                         contourIdx=-1,
                         color=border_color,
                         thickness=2,
                         lineType=open_cv.LINE_8)
    moments = open_cv.moments(coordinates)

    center = (int(moments["m10"] / moments["m00"]) - 3,
              int(moments["m01"] / moments["m00"]) + 3)

    open_cv.putText(image,
                    label,
                    center,
                    font,
                    font_scale,
                    font_color,
                    line_thickness,
                    open_cv.LINE_AA)
```

### 7.1.4 coordinates generator

```
import cv2 as open_cv
import numpy as np

from colors import COLOR_WHITE
from drawing_utils import draw_contours


class CoordinatesGenerator:
    KEY_RESET = ord("r")
    KEY_QUIT = ord("q")

    def __init__(self, image, output, color):
        self.output = output
        self.caption = image
        self.color = color

        self.image = open_cv.imread(image).copy()
        self.click_count = 0
        self.ids = 0
        self.coordinates = []

        open_cv.namedWindow(self.caption, open_cv.WINDOW_GUI_EXPANDED)
        open_cv.setMouseCallback(self.caption, self.__mouse_callback)

    def generate(self):
        while True:
            open_cv.imshow(self.caption, self.image)
            key = open_cv.waitKey(0)

            if key == CoordinatesGenerator.KEY_RESET:
                self.image = self.image.copy()
            elif key == CoordinatesGenerator.KEY_QUIT:
                break
        open_cv.destroyWindow(self.caption)

    def __mouse_callback(self, event, x, y, flags, params):

        if event == open_cv.EVENT_LBUTTONDOWN:
            self.coordinates.append((x, y))
            self.click_count += 1

            if self.click_count >= 4:
                self.__handle_done()

            elif self.click_count > 1:
                self.__handle_click_progress()

        open_cv.imshow(self.caption, self.image)

    def __handle_click_progress(self):
        open_cv.line(self.image, self.coordinates[-2],
        self.coordinates[-1], (255, 0, 0), 1)

    def __handle_done(self):
        open_cv.line(self.image,
                     self.coordinates[2],
                     self.coordinates[3],
                     self.color,
                     1)
```

```
open_cv.line(self.image,
             self.coordinates[3],
             self.coordinates[0],
             self.color,
             1)

self.click_count = 0

coordinates = np.array(self.coordinates)

self.output.write("-\n          id: " + str(self.ids) + "\n
coordinates: [" +
                  "[" + str(self.coordinates[0][0]) + "," +
                  str(self.coordinates[0][1]) + "]," +
                  "[" + str(self.coordinates[1][0]) + "," +
                  str(self.coordinates[1][1]) + "]," +
                  "[" + str(self.coordinates[2][0]) + "," +
                  str(self.coordinates[2][1]) + "]," +
                  "[" + str(self.coordinates[3][0]) + "," +
                  str(self.coordinates[3][1]) + "]]\n")

draw_contours(self.image, coordinates, str(self.ids + 1),
COLOR_WHITE)



for i in range(0, 4):
    self.coordinates.pop()

self.ids += 1
```

### 7.1.5 colors

```
COLOR_BLACK = (0, 0, 0)
COLOR_BLUE = (0, 0, 255)
COLOR_GREEN = (0, 255, 0)
COLOR_RED = (255, 0, 0)
COLOR_WHITE = (255, 255, 255)
```

## 7.2 Dart

### 7.2.1 Main

```
import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

var colors = new List(96);
void main() {
  runApp(new MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Generated App',
      theme: new ThemeData(
        brightness: Brightness.dark,
        fontFamily: 'Roboto',import 'package:flutter/material.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

var colors = new List(96);
```

```
void main() {
  runApp(new MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Generated App',
      theme: new ThemeData(
        brightness: Brightness.dark,
        fontFamily: 'Roboto',
      ),
      home: new MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key}) : super(key: key);

  @override
  _MyHomePageState createState() => new _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
```

```
return new Scaffold(
    appBar: new AppBar(
      centerTitle: true ,
      title: new Text('Smart Parking'),
    ),
    body: SingleChildScrollView(
    child: StreamBuilder(
      stream: Firestore.instance.collection('Lot7').snapshots(),
      builder: (context, snapshot) {
        if (!snapshot.hasData)
          return Text('Loading Data... Please Wait...');
        for(var i = 0; i < 96; i++) {
          if (snapshot.data.documents[i]['Open'] == "False") {
            colors[i] = Colors.red;
          }
          else if (snapshot.data.documents[i]['Open'] == "True"){
            colors[i] = Colors.green;
          }
        }

        return Column(
            children: <Widget>[

        new Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
        new RaisedButton(
        onPressed: changeColor,
        padding: const EdgeInsets.all(8.0),
textColor: Colors.white,
color: colors[64],
child: new Text("7B 1"),
),
new RaisedButton(
onPressed: changeColor,
textColor: Colors.white,
color: colors[32],
padding: const EdgeInsets.all(8.0),
child: new Text(
"7A 33",
),
),
new RaisedButton(
  onPressed: changeColor,
  padding: const EdgeInsets.all(8.0),
  textColor: Colors.white,
  color: colors[0],
  child: new Text("7A 1"),
),
],
),

new Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    new RaisedButton(
      onPressed: changeColor,
      padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[65],
          child: new Text("7B 2"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[33],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 34",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[1],
          child: new Text("7A 2"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[66],
          child: new Text("7B 3"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[34],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 35",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[2],
          child: new Text("7A 3"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[67],
          child: new Text("7B 4"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[35],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 36",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[3],
          child: new Text("7A 4"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[68],
          child: new Text("7B 5"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[36],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 37",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[4],
          child: new Text("7A 5"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[69],
          child: new Text("7B 6"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[37],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 38",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[5],
          child: new Text("7A 6"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[70],
          child: new Text("7B 7"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[38],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 39",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[6],
          child: new Text("7A 7"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[71],
          child: new Text("7B 8"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[39],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 40",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[7],
          child: new Text("7A 8"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[72],
          child: new Text("7B 9"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[40],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 41",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[8],
          child: new Text("7A 9"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[73],
          child: new Text("7B 10"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[41],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 42",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[9],
          child: new Text("7A 10"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[74],
          child: new Text("7B 11"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[42],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 43",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[10],
          child: new Text("7A 11"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[75],
          child: new Text("7B 12"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[43],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 44",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[11],
          child: new Text("7A 12"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[76],
          child: new Text("7B 13"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[44],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 45",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[12],
          child: new Text("7A 13"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[77],
          child: new Text("7B 14"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[45],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 46",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[13],
          child: new Text("7A 14"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[78],
          child: new Text("7B 15"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[46],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 47",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[14],
          child: new Text("7A 15"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[79],
          child: new Text("7B 16"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[47],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 48",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[15],
          child: new Text("7A 16"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[80],
          child: new Text("7B 17"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[48],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 49",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[16],
          child: new Text("7A 17"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[81],
          child: new Text("7B 18"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[49],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 50",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[17],
          child: new Text("7A 18"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[82],
          child: new Text("7B 19"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[50],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 51",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[18],
          child: new Text("7A 19"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[83],
          child: new Text("7B 20"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[51],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 52",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[19],
          child: new Text("7A 20"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[84],
          child: new Text("7B 21"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[52],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 53",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[20],
          child: new Text("7A 21"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[85],
          child: new Text("7B 22"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[53],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 54",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[21],
          child: new Text("7A 22"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[86],
          child: new Text("7B 23"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[54],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 55",
          )
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[22],
          child: new Text("7A 23"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[87],
          child: new Text("7B 24"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[55],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 56",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[23],
          child: new Text("7A 24"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[88],
          child: new Text("7B 25"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[56],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 57",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[24],
          child: new Text("7A 25"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[89],
          child: new Text("7B 26"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[57],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 58",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[25],
          child: new Text("7A 26"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[90],
          child: new Text("7B 27"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[58],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 59",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[26],
          child: new Text("7A 27"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[91],
          child: new Text("7B 28"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[59],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 60",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[27],
          child: new Text("7A 28"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[92],
          child: new Text("7B 29"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[60],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 61",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[28],
          child: new Text("7A 29"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[93],
          child: new Text("7B 30"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[61],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 62",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[29],
          child: new Text("7A 30"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[94],
          child: new Text("7B 31"),
        ),
        new RaisedButton(
          onPressed: changeColor,
          textColor: Colors.white,
          color: colors[62],
          padding: const EdgeInsets.all(8.0),
          child: new Text(
            "7A 63",
          ),
        ),
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
          color: colors[30],
          child: new Text("7A 31"),
        ),
      ],
    ),
    new Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: <Widget>[
        new RaisedButton(
          onPressed: changeColor,
          padding: const EdgeInsets.all(8.0),
          textColor: Colors.white,
            color: colors[95],
            child: new Text("7B 32"),
          ),
          new RaisedButton(
            onPressed: changeColor,
            textColor: Colors.white,
            color: colors[63],
            padding: const EdgeInsets.all(8.0),
            child: new Text(
              "7A 64",
            ),
          ),
          new RaisedButton(
            onPressed: changeColor,
            padding: const EdgeInsets.all(8.0),
            textColor: Colors.white,
            color: colors[31],
            child: new Text("7A 32"),
          ),
        ],
      ),




      ]
      ,
      );
    },
              )));
      }
}

void buttonPressed() {}

void changeColor() {}
```