**CBL project final product backlog – Ron Peer and Lavi Batzia – “mapPIN”**

Introduction:

Our proposed game is a geographic locator quiz, where you choose between a few modes of play

1. Find a country on the world map
2. Find a city on the world map

Topics of choice:

1. File management (accessing image files, storing and accessing data)
2. Git (Version Control)

Backlog Items (sorted by development significance):

**Name:** Data storage and access

**How to demo:** files containing space delimited data. Data includes city names, locations, flag image file locations and country. During program execution, necessary data will be queried from the relevant files.

**Name:** Render game screen

**How to demo:** Load the 4 elements to the new window – the world map as an image, a timer, the name of the current player, and the name of the location to find. Can be resized when window resolution is changed.

**Name:** Mouse object (implemented in the map screen)

**How to demo:** During the game, a click on the screen will stop the timer and initiate score calculation with the coordinates of the clicking action.

**Name:** Start (mode selection) Screen

**How to demo:** Render a screen with 2 text areas for the players’ names and radio buttons representing the different game modes, as well as a start button. Until the start button is pressed, the players can freely edit the name and pick a mode.

Pressing the start button will transfer you to the main game screen according to the last values.

**Name**: Transfer Turns Between players

**How to demo**: Once a player has made their choice and the score has been calculated, it is shown on the and screen and then after a pause a new game starting state is loaded for the next player in line. New player’s name will be reflected on the screen.

**Name:** Timer object (Action Listener)

**How to demo:** Tracks the time passed in turns and pauses. When triggered by a mouse click event, stops the countdown.

**Name:** Image analysis – country mode

**How to demo:** When a click event of the mouse is caught in this game mode, the coordinates of the mouse along with the time interval from the timer will be used to identify on which country the mouse has landed on and calculate the score. If it matches the clue, points will be awarded based on speed of response. If missed, no points will be awarded. The score will be sent to be displayed on the intermediatory screen.

**Name:** Distance calculation – city mode

**How to demo:** When an event of the mouse is caught in this game mode, the coordinates of the mouse along with the time interval from the timer will be used to calculate distance between the mouse and city of the round and calculate the score. Points will be awarded based on proximity to the clue and speed of response. The score will be sent to be displayed on the intermediatory screen.

**Name:** Game over screen

**How to demo:** After all rounds are over the last screen loads automatically containing a overall scores for both players.

**Name:** player object

**How to demo:** At the start of the game, each player could choose their name. During each of the players turns in the game, no matter the mode, scores will be added to the list of scores of each player. At the end screen of the game, the overall score of each player will be displayed.

**Integrated learning concepts:**

**File management:** Data will be read from the files – cities name file, images of flag, coordinates of cities. Data will also include file locations necessary for the game to be played, or locations to create files if necessary.

**Version Control:**

Using the resource website along with the commands practice website, we learned basic and all necessary commands in order to be able to collaboratively work on the same project and track changes. Using branching, each one of us was able to focus on their on part while still updating the other, using push-pull commands to the origin. During the learning process, we explored the previous commits we made due to changes and bugs. We merged into the main branch a few times when full tested code was finished. As part of the development, we frequently switched branches to learn, test, improve and integrate each other’s work.

Commands used:

* git commit (-a -m)
* git add
* git branch
* git status
* git push origin
* git fetch origin
* git pull origin
* git checkout (-f)
* git merge
* git stash
* git log (q to exit)
* git stash pop