*Platform Game*

Objective & Theme Layout:

Objective: Creating a platformer game that has a character traveling through multiple levels alongside obstacles with goals required to pass to the next level.

Theme: In our platformer game, the player controls a camper to jump, fly, or drive between suspended platforms and collect hidden items in order to complete the level. Each level features a different background within the level the player must traverse the terrain by jumping. The player often has some control over when the jumps are in order to avoid letting their character miss necessary jumps.

SCENARIOS

1. The first scenario takes place at the character’s home. The background is an animated image of a home. The image is placed in the google drive before being coded in the game. Our code will create lines, rectangles and triangles will be strategically placed on the line of the uploaded image. The character will jump and land on the lines, rectangles and triangles of the scenery. One or more items will be randomly placed throughout the background. Retrieving all the items is the only way the character can move on to the next level.

* In the first scenario the character will have to retrieve a bag and key.
* Once the character retrieves both items, it is allowed to continue to the right of the screen to continue to the next level.

1. The second scenario has an animated airport background. The character is in the airplane. So, the airplane is more visible than the character. Now, the airplane has to stay up in the air while trying to get the one item of the level. The one item is the airplane ticket. Once the ticket is retrieve, the airplane will have access to continue to the right of the screen.
2. The last scenario is an animated hallway where the character has to grab the key to continue to the right of the screen.

If the items aren’t grabbed in 2 mins, the level will reset. Once all the items from all three levels are retrieved, the player wins.

Group Assignments

Christall

* Background images: utilizing background()
* Documenting updates

Marcedes-Antoniya

* Each of the rooms are laid out according to our sketches
* Formal Presentation Powerpoint & Script

Solomon

* Initiating code: Implementing the game functions by having the character move, obtain items, power ups?
* Figuring out coding jumps: making variables of speed, position and acceleration to make jumps accurate.

Will

* Aligning platforms on the images using the rect function
* Vertical lines to jump on using the line function

**Plan of Action**

Step 1: create window

Step 2: load background image

Step 3: create platforms in processing

Step 4: create character

Step 5: assign key press functions

Step 6:

Project Examples

<https://youtu.be/c9ZxyS2vyiM>

<https://youtu.be/spz5I6P0aNg>

Platformer game: <https://www.openprocessing.org/sketch/539124>

Platformer jump tester: <https://www.openprocessing.org/sketch/86786>

<https://www.openprocessing.org/sketch/514105>

<https://www.openprocessing.org/sketch/554613>

**New Tasks:**

* Continuous run 5 levels per background pic
* Each time you get off the screen restart back at the beginning
* Obstacles through Y-axis and/or X-axis
* Create boundaries
* Create a character
* When 5 levels pass move to world 2 and change pic

def setup():

size(922,615)

img = loadImage('level1-house.jpg')

background(0, 255, 0)

global rect\_x, rect\_y, speedX, speedY, img

rect\_x = 300

rect\_y = 490

speedX = 5

speedY= 5

def draw():

# global rect\_x, rect\_y, speedX, speedY, img

# image(img, 0, 0)

# rect(rect\_x, rect\_y, 25, 25)

# # if rect\_y == 440:

# # speedY = -speedY

# # if rect\_y == 560:

# # speedY = -speedY

# if key == 'w' or key == 'W':

# rect\_y -= speedY

# elif key == 's' or key == 'S':

# rect\_y += speedY

# elif key == 'a' or key == 'A':

# rect\_x -= speedX

# elif key == 'd' or key == 'D':

# rect\_x += speedX

# elif key == 'q' or key == 'Q':

# rect\_y = rect\_y

# rect\_x = rect\_x

rect(66,605,850,10)

rect(63,500,10,120)

rect(910,450,10,150)

rect(450,270,320,10)

rect(200,440,325,10)

rect(600,440,310,10)

rect(510, 280, 10,80)

rect(200, 280, 10, 170)

rect(200, 440, 10, 80)

rect(630, 440, 10, 100)

quad(488, 110, 218, 267, 229, 266, 495, 114)

quad(771, 272, 500, 120, 494, 128, 752, 268)

# quad(776, 275, 779, 307, 785, 275, 784, 305)

print mouseX,mouseY