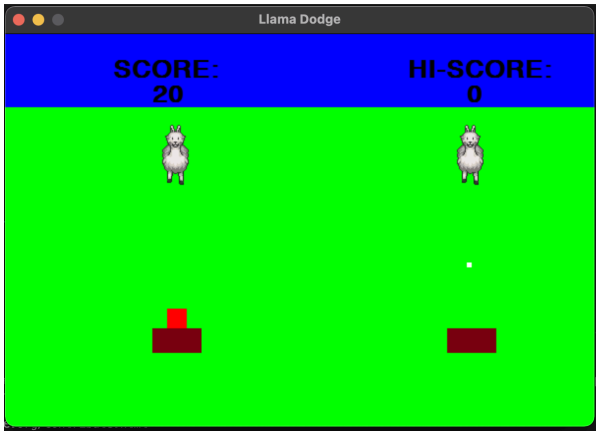
Llama Dodge

By: Kartheek Chinta



Design:

This game (which I named Llama Dodge) may look simple, but supported many features, which were created with the express intent to make the different aspects in this game possible.

The theme of this project was llamas, so I decided to implement the spit of llamas into the project. I made the llamas the enemies of the game, whose spit you were trying to escape from. The llamas would spit at you and you would dodge. This would continue until you died and the goal of the game was to survive as long as possible.

To begin to describe the aspects of the game, we must first look at the different sprites in the game. I will use UML Class diagrams and UML State Diagrams.

Boundary Class Diagram:

Boundary
Attributes: rect.x: int rect.y: int colour: string width: int height: int
Behaviours: drawToScreen(): void update(): void

Boundary State Diagrams:

Boundary	upBoundary
Colour	green
width	600
height	10
rect.x	0
rect.y	75

Boundary	downBoundary
Colour	green
width	600
height	10
rect.x	0
rect.y	350

Boundary	leftBoundary
Colour	green
width	5
height	400
rect.x	-25
rect.y	0

Boundary	rightBoundary
Colour	green
width	5
height	400
rect.x	620
rect.y	0

Llama Class Diagram:

Llama
Attributes: image: void rect.x: int rect.y: int
Behaviours: drawToScreen(): void update(): void

Llama State Diagram:

Llama	llama
image	Llama.png
rect.y	75
rect.x	Depends on # of llamas

Spit Class Diagram:

Spit
Attributes: width: int height :int rect.x: int rect.y: int spit.changeY:int
Behaviours: drawToScreen(): void update(): void deletion(): void killPlayer(): void

Spit State Diagram:

Spit	spit
width	5
height	5
rect.x	llama.rect.x
rect.y	llama.rect.y
changeY	spitSpeed

Platform Class Diagram:

Platform
Attributes: width: int height: int rect.x: int rect.y: int
Behaviours: drawToScreen(): void update(): void

Platform State Diagram:

Platform	platform
width	50
height	25
rect.x	Depends on # of platforms
rect.y	300

Player Class Diagram:

Player
Attributes: rect.x: int Rect.y: int changeX: int changeY: int changeSpeedY: int
Behaviours: drawToScreen(): void update(): void move(keyboard interaction): void deletion(): void

Player State Diagram:

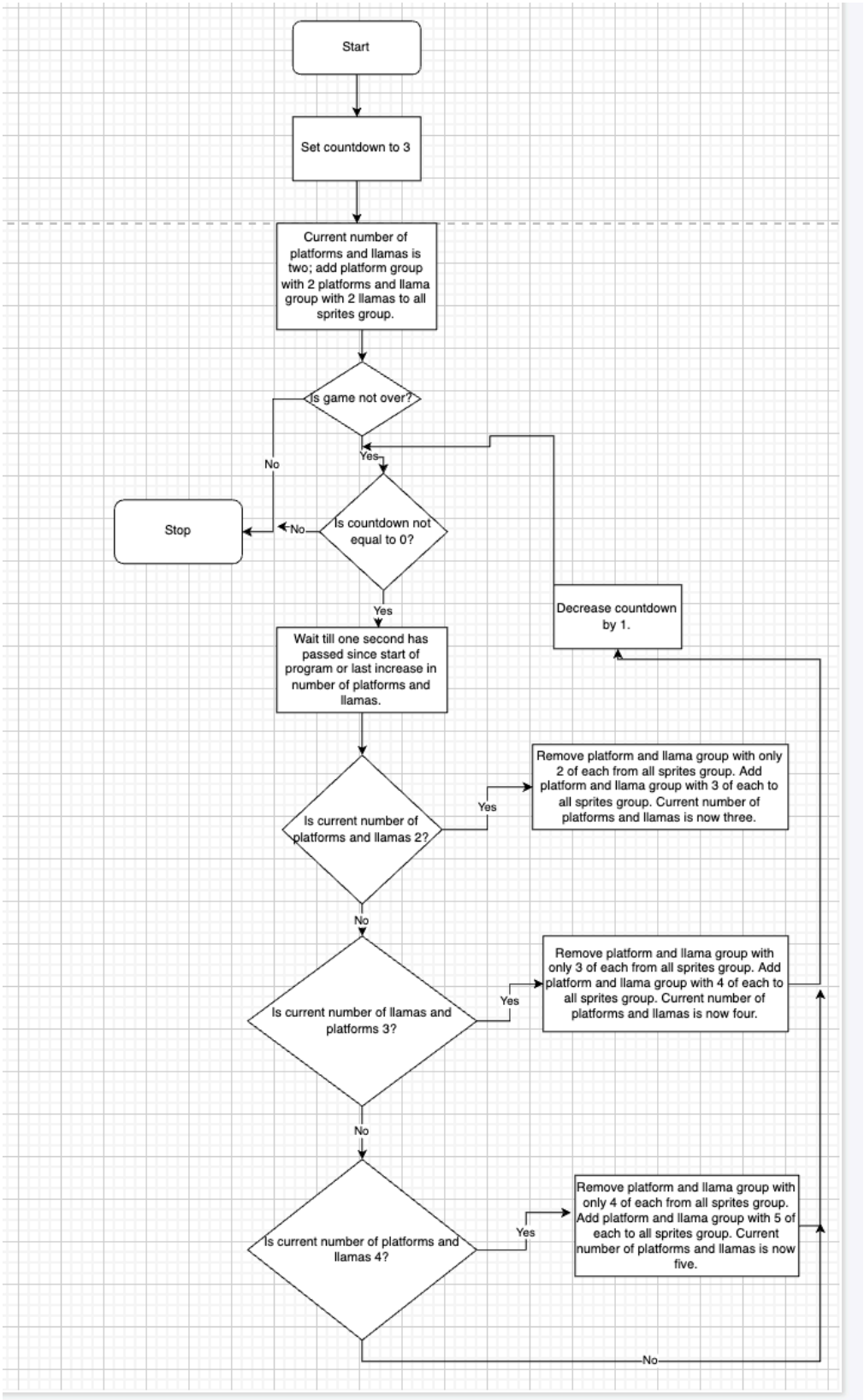
Player	player
rect.x	165
rect.y	280
changeX	0
changeY	0
changeSpeedY	0

Flowcharts:

The next aspect of the project was the platforms and llamas. I did not just want to create some platforms and llama that would stay the same throughout the game. So I made it so that the number of llamas and platforms and llamas would increase as the game went. The first thing I did was create four different platform groups, through a series of for loops. I could make a subprogram or one for loop out of this as the spacing was a little tricky with different numbers of platforms and would only work when I made the groups individually.

Increase number of platforms and llamas flow chart (after I created all four groups of platforms and llamas) on next page:

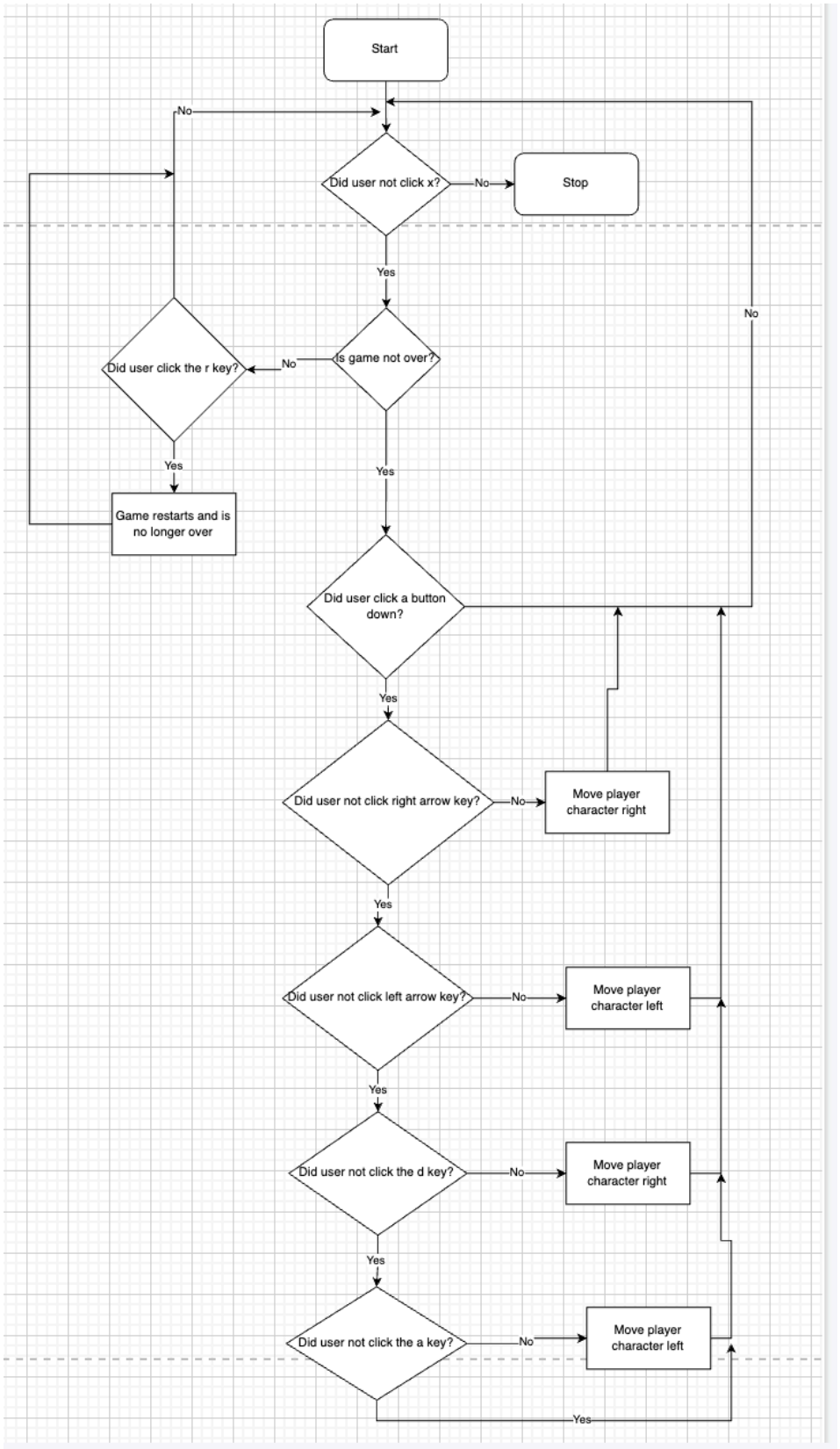
PlatformLlamaNumberIncrease(User Event)


Start
Set countdown to 3
Current number of platforms and llamas is two; add platform group with 2 platforms and llama group with 2 llamas to all sprites group.
Is game not over?
No
Yes
Stop
No
Is countdown not equal to 0?
Yes
Wait till one second has passed since start of program or last increase in number of platforms and llamas.
Decrease countdown by 1.
Is current number of platforms and llamas 2?
Yes
Remove platform and llama group with only 2 of each from all sprites group. Add platform and llama group with 3 of each to all sprites group. Current number of platforms and llamas is now three.
No
Is current number of llamas and platforms 3?
Yes
Remove platform and llama group with only 3 of each from all sprites group. Add platform and llama group with 4 of each to all sprites group. Current number of platforms and llamas is now four.
No
Is current number of platforms and llamas 4?
Yes
Remove platform and llama group with only 4 of each from all sprites group. Add platform and llama group with 5 of each to all sprites group. Current number of platforms and llamas is now five.
No

The next main aspect of the project was the player movement and user interaction. This part was pretty simple.

Flowchart for user interaction on next page:

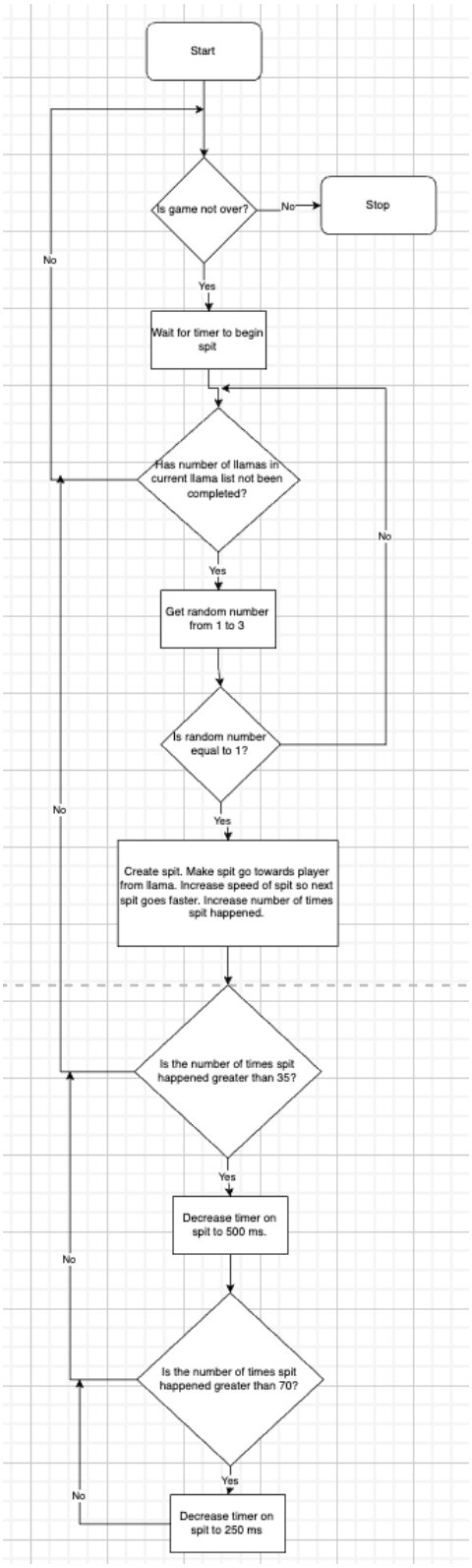
UserInteraction(Main event loop)


Start
No
Did user not click x?
No
Stop
Yes
No
Did user click the r key?
No
Is game not over?
Yes
Yes
Game restarts and is no longer over
Yes
Did user click a button down?
Yes
Did user not click right arrow key?
No
Move player character right
Yes
Did user not click left arrow key?
No
Move player character left
Yes
Did user not click the d key?
No
Move player character right
Yes
Did user not click the a key?
No
Move player character left
Yes

The final main aspect of the project was to create the spit and make it go from the llama downwards.

Flowchart to create spit and send towards platform on next page:

SpitCreation(User Event + 1)



One last aspect was the collision. This can be explained without the flowchart. I basically used sprite collision to check if the player character landed on a platform, if the player hit a spit

sprite, if the player hit a boundary, or if the spit hit the boundary. If the player hit the spit or boundary, it would be game over. If the spit hit a boundary, it would disappear. If the player hit a platform, they would be teleported on top of it to the middle.

There was an aspect that I wanted to implement that I was not able to. These are power ups. One power up idea that I had was a platform that would temporarily protect you from the spit. The idea for this was that the player would randomly (1 in 50 chance) get a platform above their heads. This platform would follow them and stay above them and would protect them for 3 seconds before disappearing. I was not able to implement this. A basic idea I had was to make the rect.x and the rect.y of the platform powerup follow the player.rect.x and the player.rect.y. When first creating the platform in a userevent, the rect.x and rect.y would be equal to the player (plus or minus a little bit) and in the game logic, I would continuously make the rect.x and rect.y equal to the player's. I would also check for collisions on this platform specifically and make spit disappear if it hit this platform. I had no other ideas for powerups.

Implementation:

The paradigm I used for this project was sprites. Each sprite was explained clearly in design so I will skip over that. The first thing I did was create a bunch of sprite groups. Then I created some boundaries for the game. These will be explained later. After that came a slightly difficult part. I had to create some lists for each number of platforms there would be in the game, but the spacing was hard to do in one for loop, so I made multiple for loops. Each for loop assigned platforms and llamas to a different group of sprites.

```
192
193  #Creating platforms and llamas for when there are 2
194  for i in range(2):
195      platform = Platform(150+300*i, 300)
196      llama = Llama(140+300*i, 75)
197      platformTwoList.add(platform)
198      llamaTwoList.add(llama)
199
200  #Creating platforms and llamas for when there are 3
201  for j in range(3):
202      platform = Platform(150+150*j, 300)
203      llama = Llama(140+150*j, 75)
204      platformThreeList.add(platform)
205      llamaThreeList.add(llama)
206
207  #Creating platforms and llamas for when there are 4
208  for k in range(4):
209      platform = Platform(120+120*k, 300)
210      llama = Llama(110+120*k, 75)
211      platformFourList.add(platform)
212      llamaFourList.add(llama)
213
214  #Creating platforms and llamas for when there are 5
215  for l in range(5):
216      platform = Platform(100+100*l, 300)
217      llama = Llama(90+100*l, 75)
218      platformFiveList.add(platform)
219      llamaFiveList.add(llama)
```

Then I added the current platform group (starts at platformTwoList and llamaTwoList) to the allSpritesGroup.

I then created a player sprite and placed it on the first platform. I then set up the game over scene and some other variables that would be used later on. I also set up pygame timers for user event and user event + 1.

```
#Setting up counter to multiply pltforms
pygame.time.set_timer(pygame.USEREVENT, 7000, True)
countDown = 3

#Setting up counter to continuously spit
pygame.time.set_timer(pygame.USEREVENT+1, 1000, True)
```

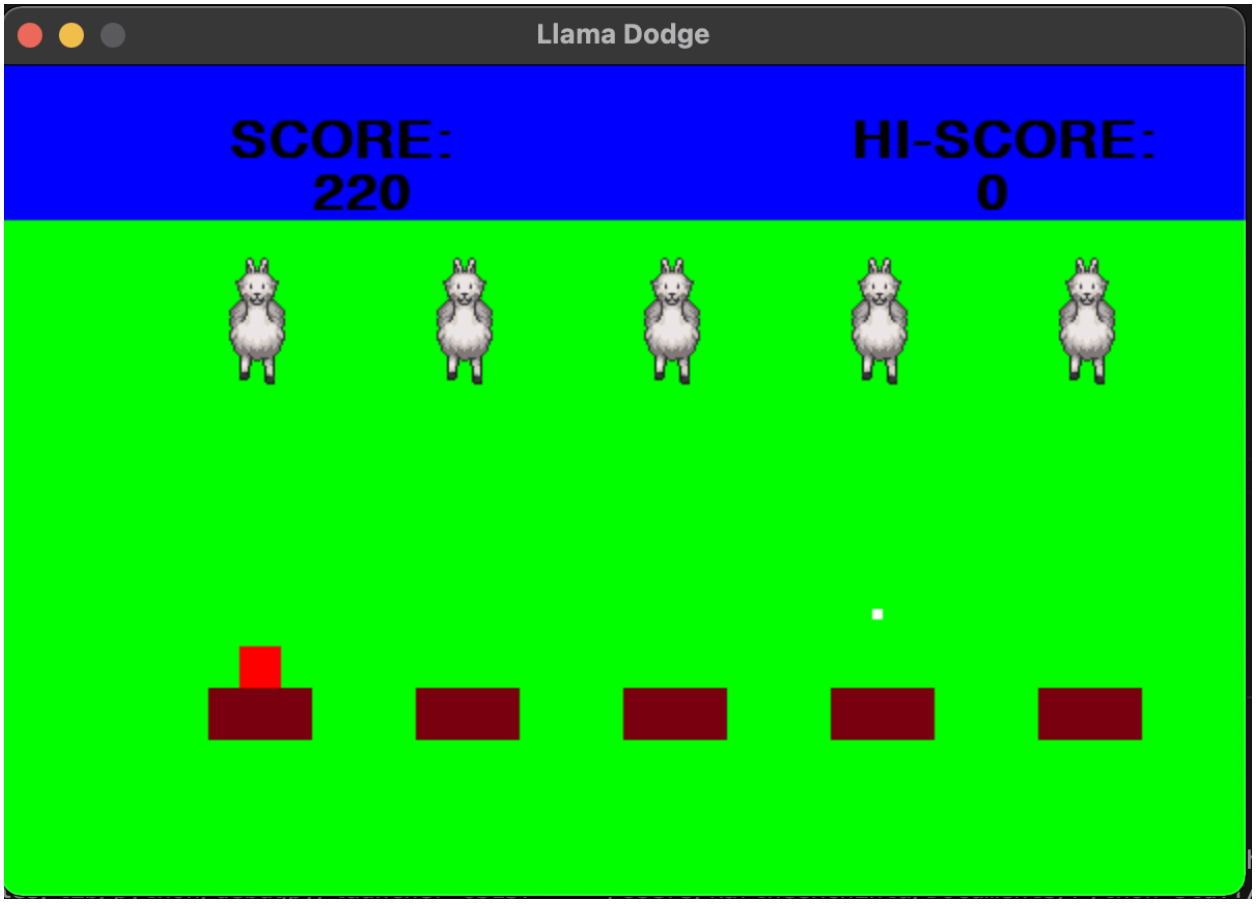
Then came the user interaction in the pygame main event loop. This was also explained clearly in design.

Then came the user events. I will first explain user event. User event was created to increase the number of platforms and llamas while the game was running. I made a series of selection statements to check how many platforms there were currently and to increase the number of platforms based on that number. The platform and llamas groups that were created earlier with for loops were used here to achieve this.

```
#Increasing number of platforms and llamas to three if there are 2 right now
if (platformCount == 2):
    platformCount += 1
    allSpritesList.remove(platformTwoList)
    allSpritesList.add(platformThreeList)
    allSpritesList.remove(llamaTwoList)
    allSpritesList.add(llamaThreeList)
    currentPlatformList.remove(platformTwoList)
    currentPlatformList.add(platformThreeList)
    currentLlamaList.remove(llamaTwoList)
    currentLlamaList.add(llamaThreeList)

#Increasing number of platforms and llamas to three if there are 3 right now
elif (platformCount == 3):
    platformCount += 1
    allSpritesList.remove(platformThreeList)
    allSpritesList.add(platformFourList)
    allSpritesList.remove(llamaThreeList)
    allSpritesList.add(llamaFourList)
    currentPlatformList.remove(platformThreeList)
    currentPlatformList.add(platformFourList)
    currentLlamaList.remove(llamaThreeList)
    currentLlamaList.add(llamaFourList)

#Increasing number of platforms and llamas to three if there are 4 right now
elif (platformCount == 4):
    platformCount += 1
    allSpritesList.remove(platformFourList)
    allSpritesList.add(platformFiveList)
    allSpritesList.remove(llamaFourList)
    allSpritesList.add(llamaFiveList)
    currentPlatformList.remove(platformFourList)
    currentPlatformList.add(platformFiveList)
    currentLlamaList.remove(llamaFourList)
    currentLlamaList.add(llamaFiveList)
```

I also changed the player position based on the number of platforms there were, so that it would spawn on top of the first platform in the group, whichever group it was.

```
#Setting up new player position every time platform and llama number increases
if (platformCount == 2):
    player.rect.x = 165
    player.rect.y = 300
elif (platformCount == 3):
    player.rect.x = 165
    player.rect.y = 300
elif (platformCount == 4):
    player.rect.x = 135
    player.rect.y = 300
elif (platformCount == 5):
    player.rect.x = 115
    player.rect.y = 300
```



I then made the countdown go down in an if statement and put the timer resent in there as well. This made it so that when the countdown hit 0, this user event would no longer take place and the counter would hit 0 when the number of platforms became 5.

We will now look at user event + 1. The timer for this one was set at 1000 milliseconds. There was no countdown (it would not stop until the game was over). I made it so that the score would increase by ten every time this event took place.

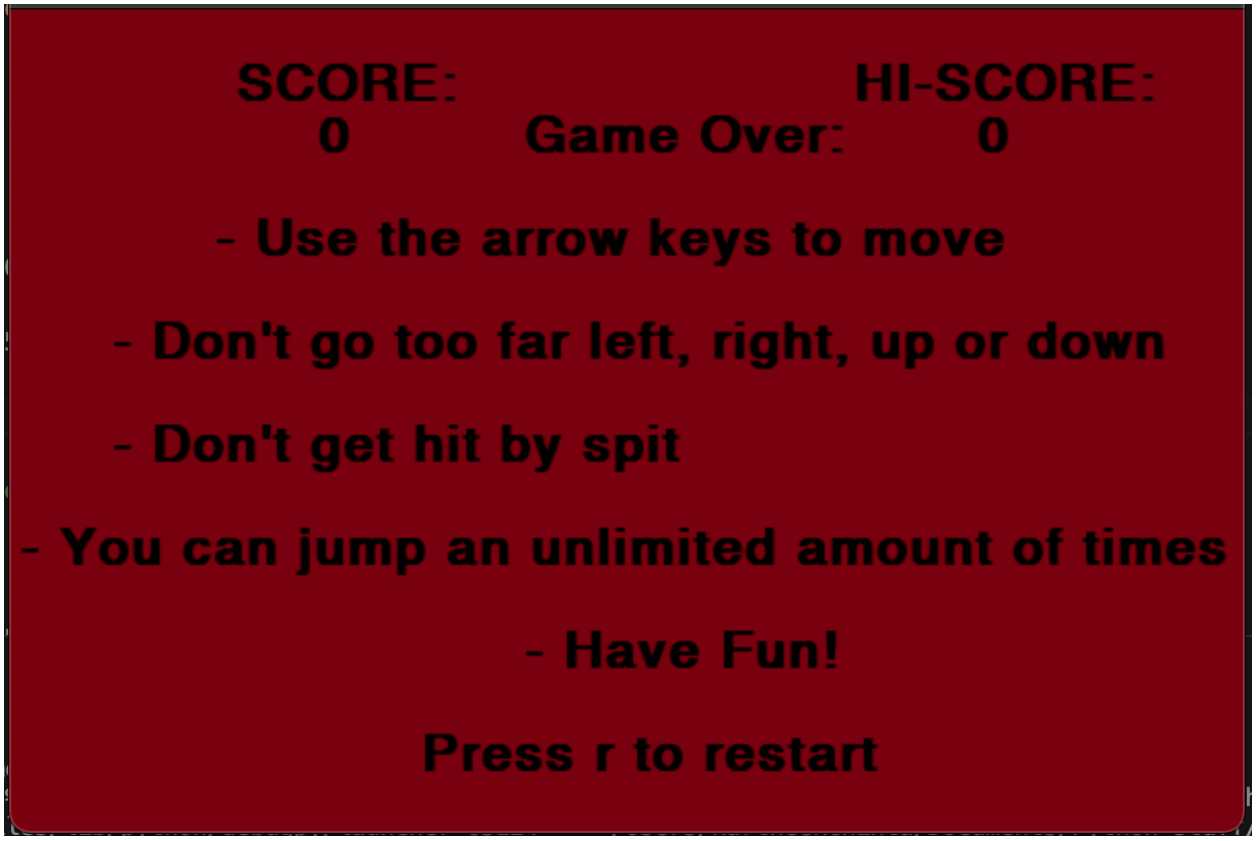
I made a for loop and went through every llama in the currentLLamaList group. I then got a random number(1, 2, or 3) and if the number was one, spit would be created for that llama and would go towards the platform underneath the llama.

```
#Going through each llama in current llama group
for llamas in currentLlamaList:

    #Choosing random number
    spitLlama = random.randrange(1, 4)

    #If random number chosen is 1 make spit come from that llama
    if (spitLlama == 1):

        #Defining spit
        spit = Spit(-5, -5)
        spit.rect.x = llamas.rect.x + 30
        spit.rect.y = llamas.rect.y + 10
        spit.changeY = spitSpeed

        #Increasing spit speed
        spitSpeed += 0.1

        #Increasing spit loop count
        spitLoopCount += 1

        #Adding spit to groups
        spitList.add(spit)
        allSpritesList.add(spit)
```

If the number of times spit was created exceeded 35, the spit time would decrease from 1000 milliseconds to 500 ms and the amount of time between each spit would decrease by 500. The spit time would halve to 250 ms again if the amount of times spit was created exceeded 70.

Then I checked if the user wanted to restart and if they did, I would restart every variable. I would empty every sprite group. I would also reset the current platform and llama groups. This would officially restart the game.

I then updated allSpritesGroup and checked for collisions which I explained in design. I then created the score text to blit to the screen. I then drew everything to the screen. I then created the screen to show when the game was over and the screen to show when the game was beginning. I then updated the score and high-score (if score was greater than high score) and blighted it to the screen. I then flipped to the screen.



Testing:

To test the program, I used many different things, including the debugger in repl.it and the debugging system in the coding software I was using. I tried changing variables to see if they were doing what I wanted them to do and to make it easier to test other aspects of the project.

I have many examples of what errors I found and how I tested the program, so I will give a few. One error was a jumping error. The jumps were not going from platform to platform and were going too far when the platform number increased above 2. This problem was an easy fix with a few if statements, but it was very,very hard to test if I was using a good number for the jumps. This was because the game was already pretty much complete by the time I got to this problem. I was proud of how it turned out, but I might have made it too hard. I successfully tested two, three and four platforms, but I was not good enough at the game to get to five platforms. I had to change the code of the game to do this, decreasing the amount of time between each platform number increase.

Another problem I had was that the spit was not being created most of the time on the first platform before. This made it easy for the player as they could sit on the first platform and

the spit rarely was created there. This was because I was using a bad, too complicated randomizing system. I realized this after playing around with it. I then came up with a much more simple, more efficient randomizer.

A big issue came with the scene change. In more demos, there is an example for scene change, but I had made a mistake. I thought I would be able to implement this after I did everything else, but as I came to it, I realized that I could not figure out how to implement it as I would need to change the code drastically. I then decided to test restarting all the variables and figured it out. This was because I was able to test many different ideas that I had.

A test case where two different components were tested together was for the boundary and the spit. I was able to test if the spit was deleted properly when it hit the boundary. This way I could test the collision, if the spit group was working properly and if the boundaries were working.

The last example of where testing helped me was in the platform and llama lists. No matter how hard I tried, I could not find out how to do it more efficiently. After testing many scenarios, like doing one platform list and llama list, one for loop for all the lists and even emptying and refilling the group, I realized that this would be the best and easiest code to use for the rest of the program.

Analysis:

In all, I think this project went well. It started as a bit of a challenge, but as I went through it, I started figuring it out more and more. As I began, I realized that I would need many different sprites for this project, so I started by creating them and adding them to lists. I wanted the platform to increase in number forever and get smaller and smaller and the player to stay wherever they landed on the platform. I then realized though that, this would be hard if not impossible for the platform increase and not smart for the player. I then decided to do just until five for the platforms. For the player, I did manage it, but I then realized a glitch. When the spit came, and if the player was on the edge of a platform, the spit went right past the,. This would make the game not fun as you could sit on an edge and never die. I then made it so that if the player hit a platform, they would be teleported to the middle of it, where the spit could hit them.

Once all of this was decided, I really started on the project. I immediately realized that I could not do the platform and llama increase efficiently and make it work the way i wanted it too. So I decided on five lists for that. The player was fairly simple and so were the boundaries. The spit gave me a little bit of trouble, but it was fixed when I made thes spit delete itself when it hit the bottom boundary. The trouble was that the spit grup got bigger and bigger and I could not

keep track of the spit. This caused glitches in the player's death and spit creation. The spit randomizer also gave me some trouble. This was also solved though.

One major problem in the project was my trip. I would lose two days due to this and had to plan my cot around it. So I decided to leave out some of the things I was planning to do and change others to be easier to code. The game over scene was hard to do, but I managed to figure it out after an hour or two. After reaching India, I had my family play the game and they had a few suggestions. There were four main ones though. The two of the four that I managed to do, was to show the score of the player after they died. I had not thought of this and did not know if I would have. The other one was to create an instructions scene at the beginning instead of only giving instructions after the user died. This was harder to do than the score, but was figured out. There were two that I was unable to do though. This was to have the user enter a name onto the screen to keep track of score and name. Realized that this would not be needed though, as the program would just forget anyways after the program stopped and I could not figure out how to have the user enter a name onto the screen. The other one was to change the player sprite and spit sprite to ellipses instead of rectangles. I tried to search it up, but was unable to do this. I was also running out of time so I decided to not do these two things.

Overall I think everything went well in the project, though I wish I had put more time and effort into making a better background and making the game more visually interesting. I would not really change anything else in the project, but I would add some things.

The part of the project that I am most proud of is the platform and llama increase and the spit. I am proud of the platform and llama increase as it was hard to figure out and yet I was still able to do. This also made the game much more interesting and fun.

If I had time, I would add power ups. These would make the game truly good and would make the user experience a lot more fun than with just dodging. I would also like to add more graphics to the program to make it more visually appealing. The game also glitches a little bit sometimes and kills you, even if the spit did not hit you and you did not hit any boundary. I was never able to figure this out as it happened rarely, but a few theories I had were that it was a problem with sprite collision or that it was a problem with my spit group. I could never test these though as it did not happen consistently. I would add a lot more if I had time, but these three things are the major things I would add or change in the program. One other thing, though not as major, would be to add sound to the game. I completely forgot about this until I started writing this report.

References:

Llama Images:
https://www.nih.gov/news-events/nih-research-matters/llama-antibody-engineered-block-coronavirus
https://www.ukri.org/news/llama-antibody-has-significant-potential-as-covid-19-treatment/

Flowchart creation:
https://app.diagrams.net/#G1HraSGt8mESG0YnesvX-fdpn6S1jj_xRi

Player sprite (all other sprites were derived from this):
http://programarcadegames.com/index.php?chapter=introduction_to_sprites&lang=en

Speed changing idea for gravity:
Mr. Reid's More Demos on repl.it

Llama sprite image:
https://opengameart.org/content/lpc-style-farm-animals

Main MVC structure of program (colours, pygame initiation, main event loop, pygame quit, etc.):
Mr. Reid's programs on repl.it

Collision Detection:
http://programarcadegames.com/index.php?chapter=introduction_to_sprites&lang=en