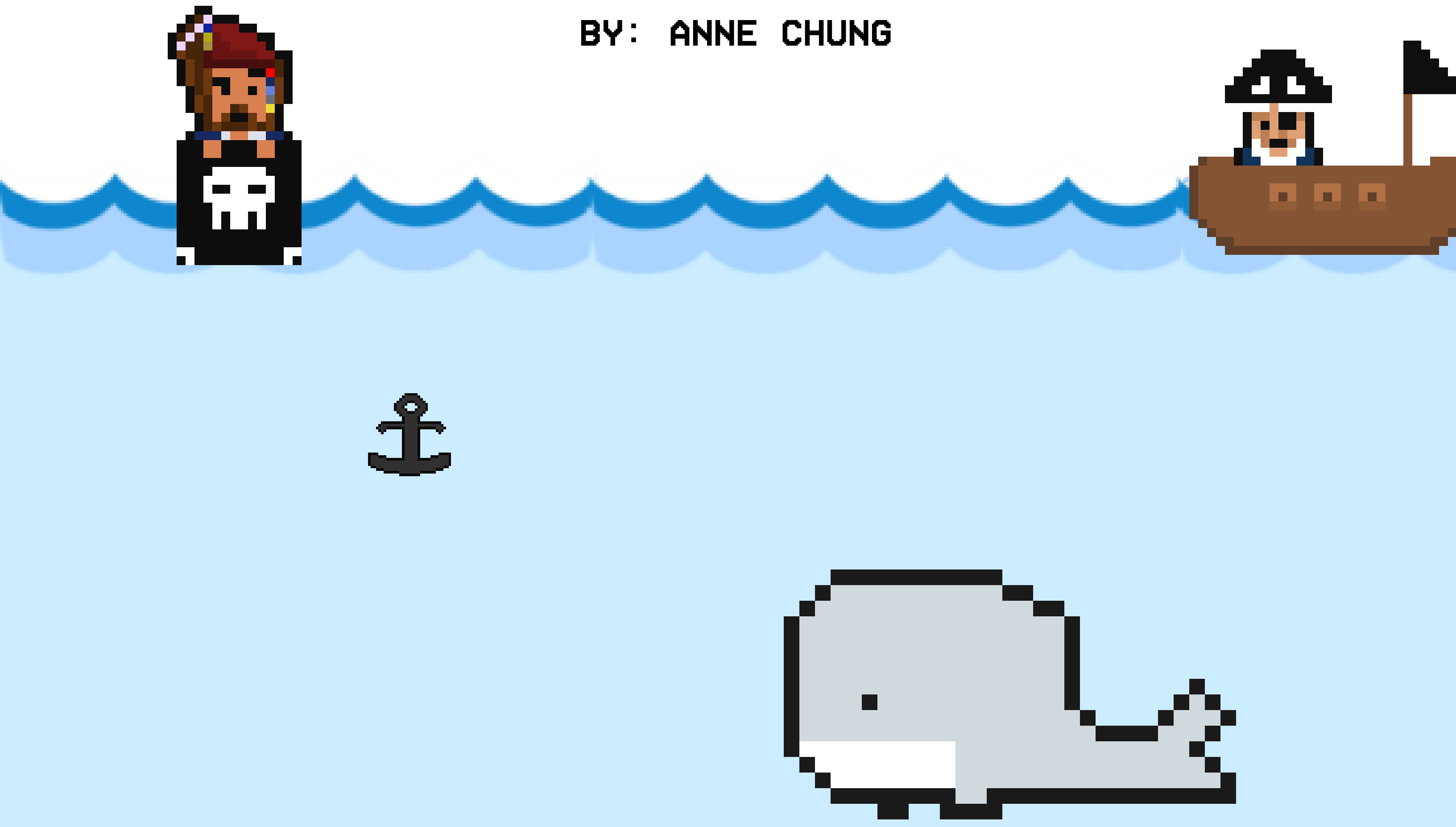
MOBY'S SICK: COMPUTER SCIENCE CPT

BY: ANNE CHUNG



Date started: Monday, June 6th, 2016

Date delivered: Friday, June 16th, 2016

Course: ICS3U1

Last updated: Friday, June 16th, 2016

Customer Requirements

The clients will be middle-school to high-school students (ages 10-18). Many teenagers in this age group play computer games. There are millions of computer games that are available to the clients, and they are continuously in the search for the perfect game.

The client needs a new game with a unique, fresh concept and an interesting storyline. This game must be simple enough so that the vast majority of students will be able to play, but complex enough so that it retains interest and attention from the users.

In order to play the game, the client will require a computer with a keyboard, which has Python and Pygame installed onto it. The client should be proficient with a computer and keyboard.

Project Timeline

	Requirements	**Design**	**Implementation**	**Testing**	**Deployment**
Proposed	June 3rd	June 4th	June 8th - 13th	June 14th - 15th	June 16th
Completed	June 3rd	June 4th to 6th	June 7th - 15th	June 15th	June 17th

The time required for implementation was greatly underestimated. Due to the prolonged implementation period, the time left for testing and maintenance was shortened. This delayed the deployment date by one day.

Design Proposal

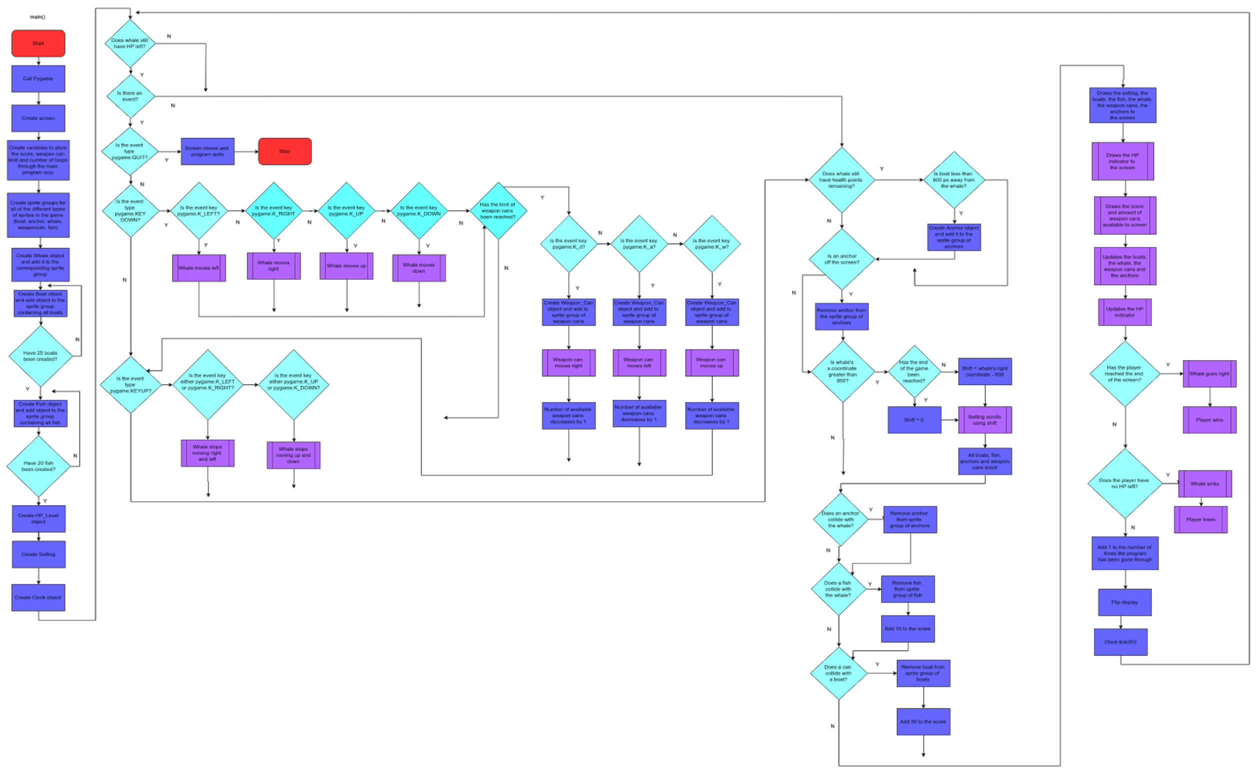
I will develop a game called “Moby’s Sick”. The storyline surrounds Moby, a small, young, whale, trying to find his way home after he has become homesick. The whale, who is the main character of the game, is controlled by the client using the arrow keys on the keyboard.

The whale is being hunted by sailors on boats, which are placed all throughout the game. They throw anchors at him to inflict damage. When an anchor hits the whale, the user will lose health points (HP). The whale is able to attack the boats using cans. The user can control where the direction the cans shoot in using the keys W, A and D on the keyboard. When a can hits a boat, the boat will disappear.

There will be fish placed throughout the game that the whale can collect to regain HP.

The goal for the user is to get from the starting point to the end point, which are indicated using red and blue flags, without losing all of the whale’s HP.

Main Program

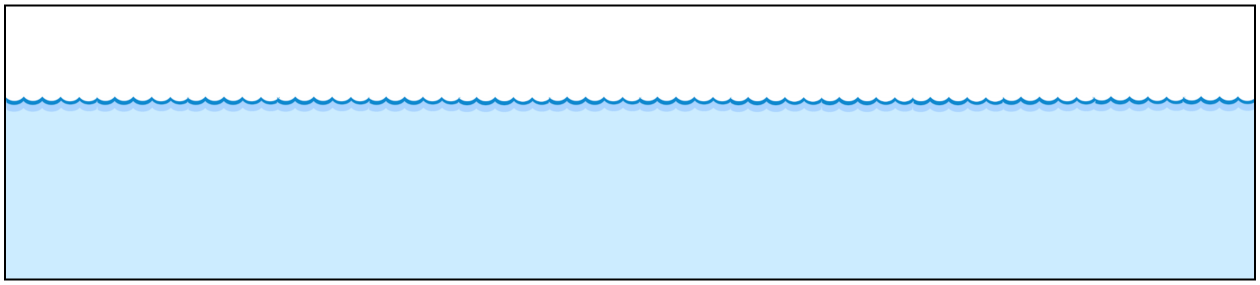

main()
Start
Call Pygame
Create screen
Create variables to store the score, weapon can limit and number of loops through the main program loop
Create sprite groups for all of the different types of sprites in the game (boat, anchor, whale, weaponcan, fish)
Create Whale object and add it to the corresponding sprite group
Create Boat object and add object to the sprite group containing all boats
Have 25 boats been created?
Create Fish object and add object to the sprite group containing all fish
Have 20 fish been created?
Create HP_Level object
Create Setting
Create Clock object
Does whale still have HP left?
Is there an event?
Is the event type pygame.QUIT?
Screen closes and program quits
Stop
Is the event type pygame.KEY DOWN?
Is the event key pygame.K_LEFT?
Whale moves left
Is the event key pygame.K_RIGHT
Whale moves right
Is the event key pygame.K_UP
Whale moves up
Is the event key pygame.K_DOWN
Whale moves down
Has the limit of weapon cans been reached?
Is the event key pygame.K_d?
Is the event key pygame.K_a?
Is the event key pygame.K_w?
Create Weapon_Can object and add to sprite group of weapon cans
Weapon can moves right
Weapon can moves left
Weapon can moves up
Number of available weapon cans decreases by 1
Is the event type pygame.KEYUP?
Is the event key either pygame.K_LEFT or pygame.K_RIGHT?
Is the event key either pygame.K_UP or pygame.K_DOWN?
Whale stops moving right and left
Whale stops moving up and down
Does whale still have health points remaining?
Is boat less than 600 px away from the whale?
Create Anchor object and add it to the sprite group of anchors
Is an anchor off the screen?
Remove anchor from the sprite group of anchors
Is whale's x-coordinate greater than 800?
Has the end of the game been reached?
Shift = whale's right coordinate - 800
Shift = 0
Setting scrolls using shift
All boats, fish, anchors and weapon cans scroll
Does an anchor collide with the whale?
Remove anchor from sprite group of anchors
Does a fish collide with the whale?
Remove fish from sprite group of fish
Add 10 to the score
Does a can collide with a boat?
Remove boat from sprite group of boats
Add 50 to the score
Draws the setting, the boats, the fish, the whale, the weapon cans, the anchors to the screen
Draws the HP indicator to the screen
Draws the score and amount of weapon cans available to screen
Updates the boats, the whale, the weapon cans and the anchors
Updates the HP indicator
Has the player reached the end of the screen?
Whale goes right
Player wins
Does the player have no HP left?
Whale sinks
Player loses
Add 1 to the number of times the program has been gone through
Flip display
Clock.tick(60)
Y
N

*if this picture is hard to see, please see “CPTmoby.png” in the CPT folder.

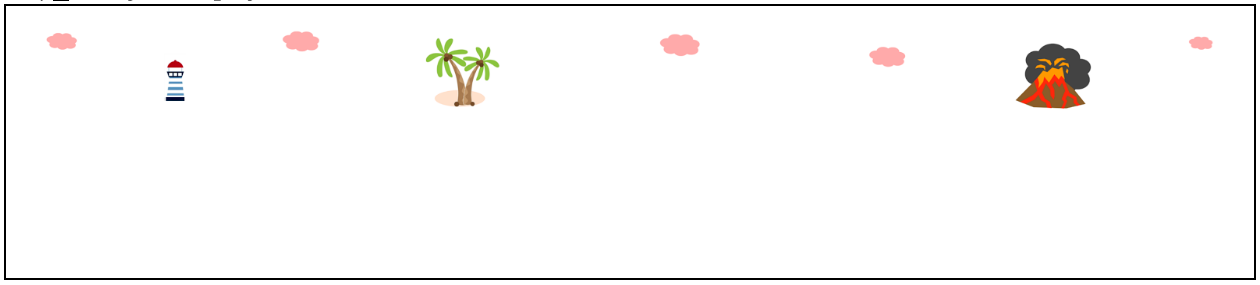
Setting (inspired by ProgramArcadeGames.com, Chapter 13)

The background images of the game will include an image of the ocean as well as an image of the surrounding sky.

“water_background.png”



“sky_background.png”



These images will be blitted to the screen, the water after the sky.
As the whale moves towards the right edge of the screen, the x – coordinates of the images will be subtracted from, so that the background will move as the whale moves further right.

Class Diagram

class Setting()
sky_background: Object() water_background: Object() start_flag: Object() end_flag: Object shift: Int limit: Int instructions: Object

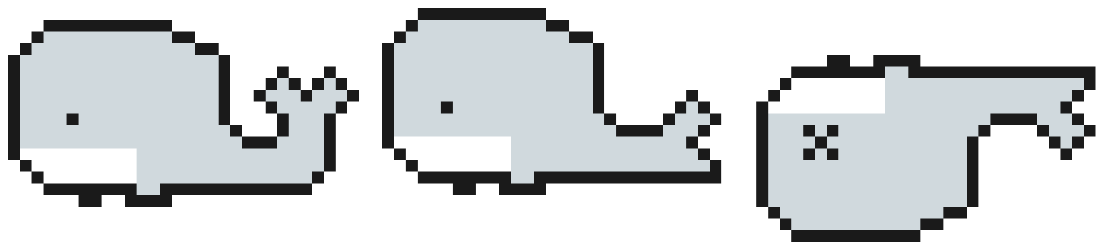
draw_water(screen)
draw_sky(screen)
update(shift)

State Diagram

setting: Setting
sky_background = pygame.image.load(“sky_background.png”).convert()
water_background = pygame.image.load(“water_background.png”.convert()
start_flag = SpriteSheet(“startflag.png”)
end_flag = SpriteSheet(“endflag.png”)
shift = 0
limit = 10500
instructions = pygame.image.load(“instructions.png”).convert()

Whale

The whale will be able to move in all directions using the arrow keys on the keyboard.
Every few frames, the whale will switch in between images on a spritesheet, to appear as if it is moving.
This is done using a spritesheet, where all of the different images are taken from, and a list which contains all of the images.
These images can be switched through easily by changing the index number that calls the list.
“Moby.png”



Class Diagram

class Whale(pygame.sprite.Sprite)
x_change: Int y_change: Int walking_frames_r: List walking_frames_r: List framechange: Int direction: String image: Object rect: Object

update (loop: int, hp_level: object)
go_right()
go_left()
go_down()
go_up()
stop_up_down()
stop_right_left()
sink()

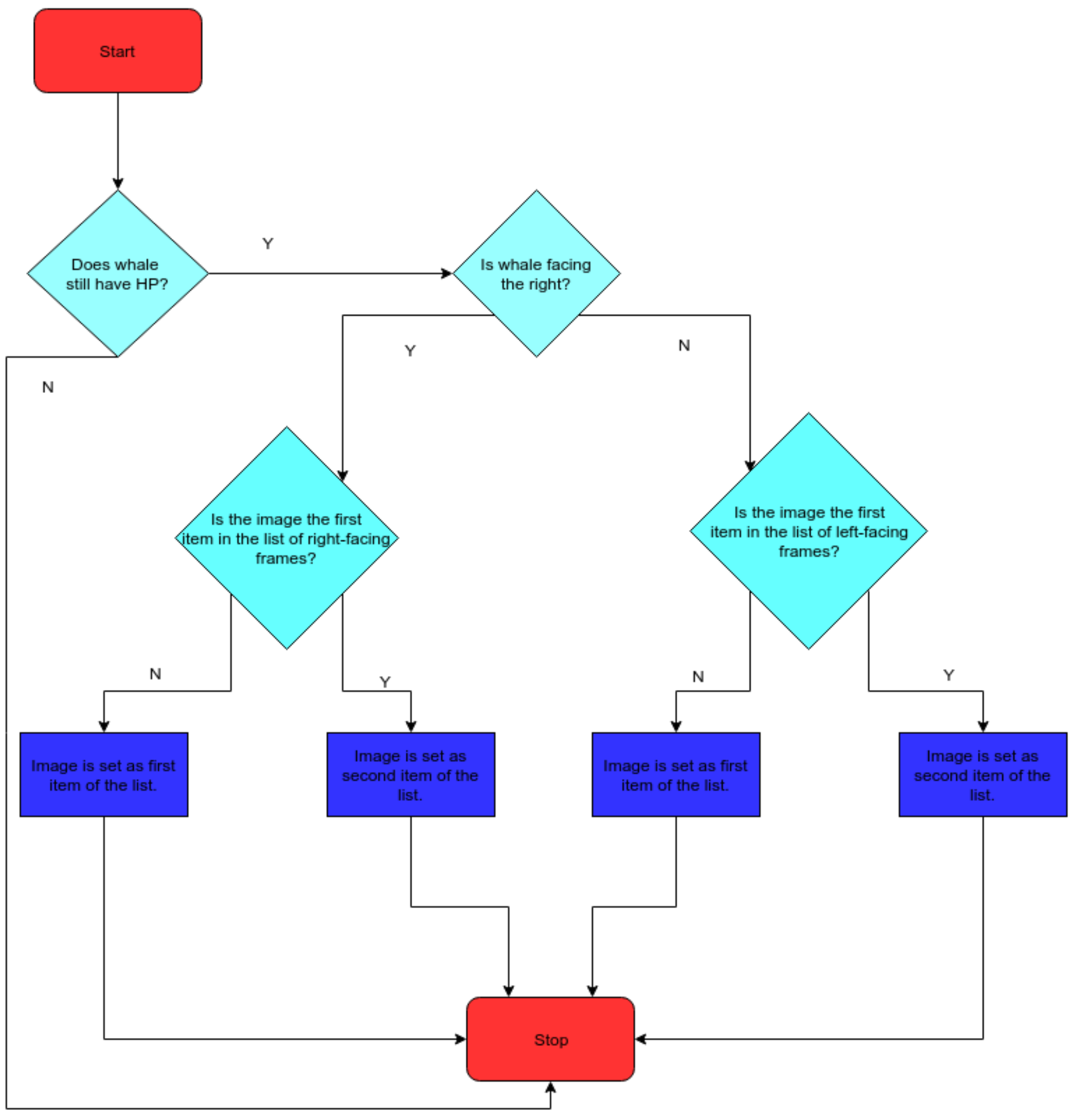
State Diagram

moby: Whale

change_x = 0
change_y = 0
walking_frames_r: [ ]
walking_frames_l: [ ]
framechange = 0
direction = “R”
image = walking_frames_r[0]

The flowchart below describes the update function of the whale, and shows how it switches in between images in a list to animate the character.

Whale.update(loop, hp_level)


Start
Does whale still have HP?
Y
N
Is whale facing the right?
Y
N
Is the image the first item in the list of right-facing frames?
Is the image the first item in the list of left-facing frames?
N
Y
N
Y
Image is set as first item of the list.
Image is set as second item of the list.
Image is set as first item of the list.
Image is set as second item of the list.
Stop

Boat

There will be 25 boats placed throughout the game, above the water. They also switch in between different images on a spritesheet to appear animated.

Class Diagram

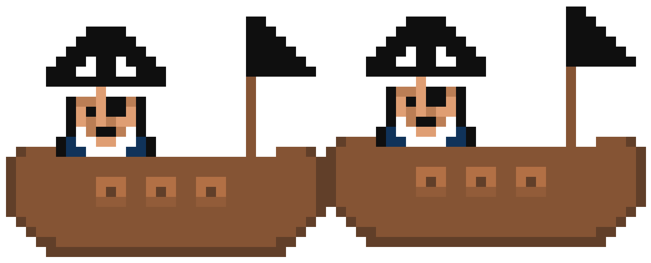
class Boat(pygame.sprite.Sprite)
speed : Int pic_index: Int walking_frames: List framechange: Int switch_speed: Int image: Object rect: Object
update(loop: Int)

State Diagram

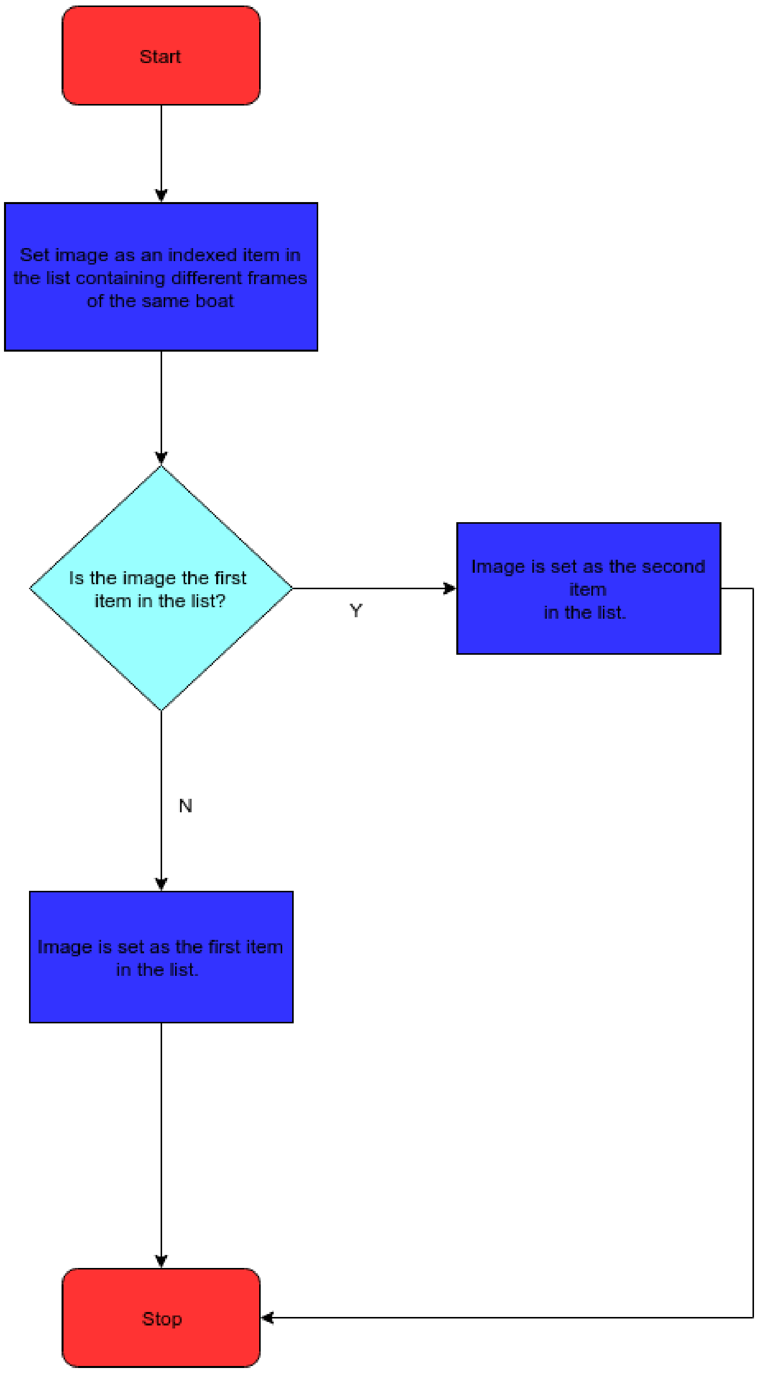
boat = Boat

speed = 30
pic_index = 2
walking_frames = [ ]
framechange = 0
switch_speed = 20
image = walking_frames[0]

“Boat1.png”



Boat.update(loop)


Start
Set image as an indexed item in the list containing different frames of the same boat
Is the image the first item in the list?
Y
Image is set as the second item in the list.
N
Image is set as the first item in the list.
Stop

Anchor

Each boat will be associated with the creation of anchors.
The coordinates of the anchor will coincide with the centre of the boat.
The anchors are created only when the player is around the boat.
Based on the coordinates of the whale, the changes made in the coordinates of the anchor will change as well.

Class Diagram

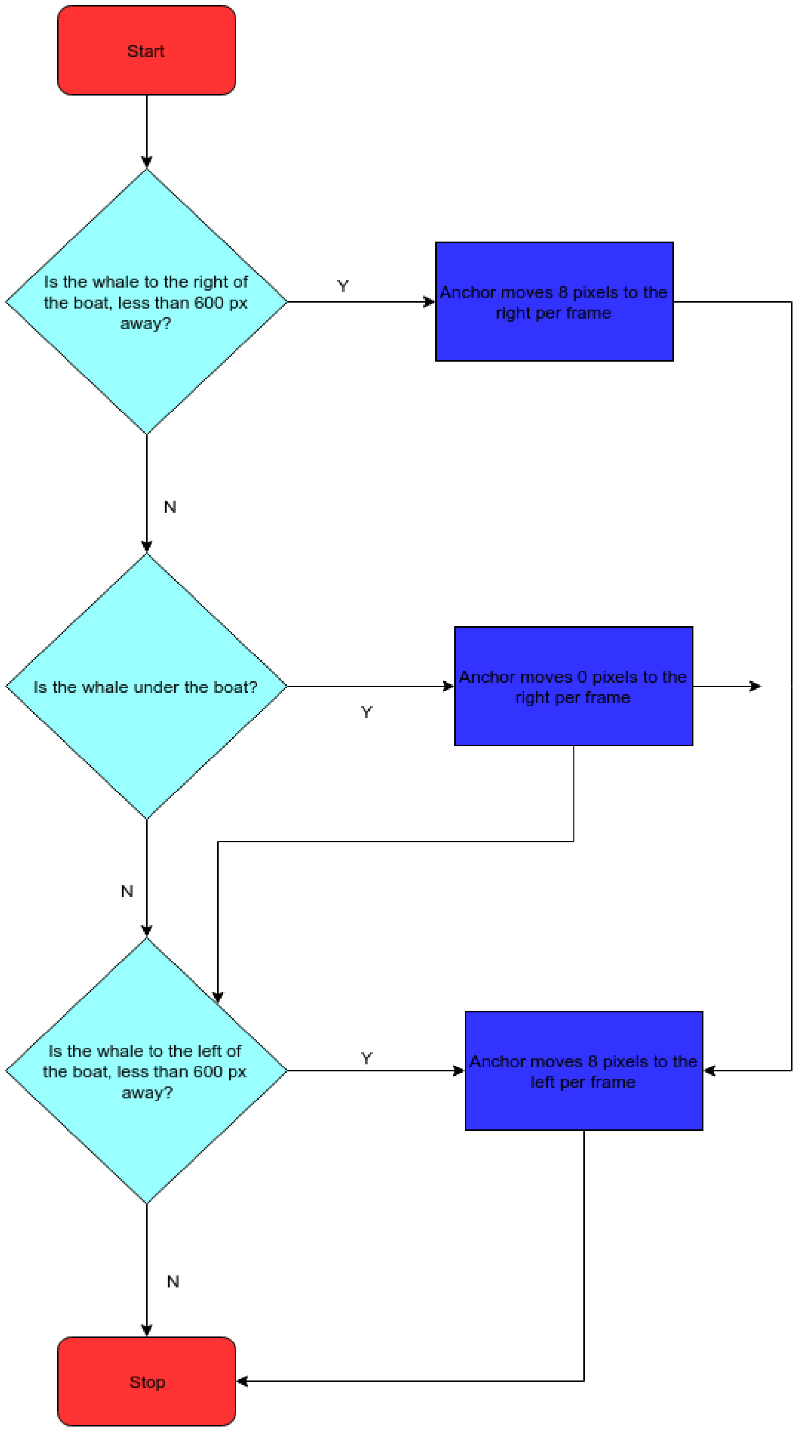
class Anchor(pygame.sprite.Sprite)
image: Object x_change: Int y_change: Int rect: Object rect.x: Object rect.y: Object
update(moby: Whale)

State Diagram

anchor = Anchor

image = anchor_image.get_image(0, 0, 32, 32)
x_change = -8
y_change = 6
rect.x = 750
rect.y = 120

Anchor.update(moby)


Start
Is the whale to the right of the boat, less than 600 px away?
Y
Anchor moves 8 pixels to the right per frame
N
Is the whale under the boat?
Y
Anchor moves 0 pixels to the right per frame
N
Is the whale to the left of the boat, less than 600 px away?
Y
Anchor moves 8 pixels to the left per frame
N
Stop

Weapon Can

The user will have a group of weapon cans to fight the boats. The cans will be created when the user presses one of the keys used to attack the boats with (A, D, W).

Class Diagram

class WeaponCan(pygame.sprite.Sprite)
image: Object x_change: Int y_change: Int rect: Object rect.x: Object rect.y: Object

update()
go_right()
go_left()
go_up()

State Diagram

weaponcan = WeaponCan

image = can_image.get_image(0, 0, 18, 19)
x_change = 5
y_change = -6
rect.x = 590
rect.y = 185

Fish

The fish are placed throughout the game, and can be collected by the whale to restore its HP. There are 20 fish in total.

Class Diagram

class Fish(pygame.sprite.Sprite)
image: Object rect: Object rect.x: Object rect.y: Object

State Diagram

fish = Fish()

image = fish_image.get_image(0, 0, 64, 30)
rect.x = 600
rect.y = 400

HP_Level

This class is responsible for determining and displaying the HP of the whale. When the whale is hit by an anchor, the HP decreases. When the whale eats a fish, the HP increases. When the HP is running low, the rectangle it is represented in will turn red, instead of its usual green.

Class Diagram

class HP_Level()
color: Object length: Int

draw(screen)
update(anchors_hit: Sprite Group, fish_collected: Sprite Group)

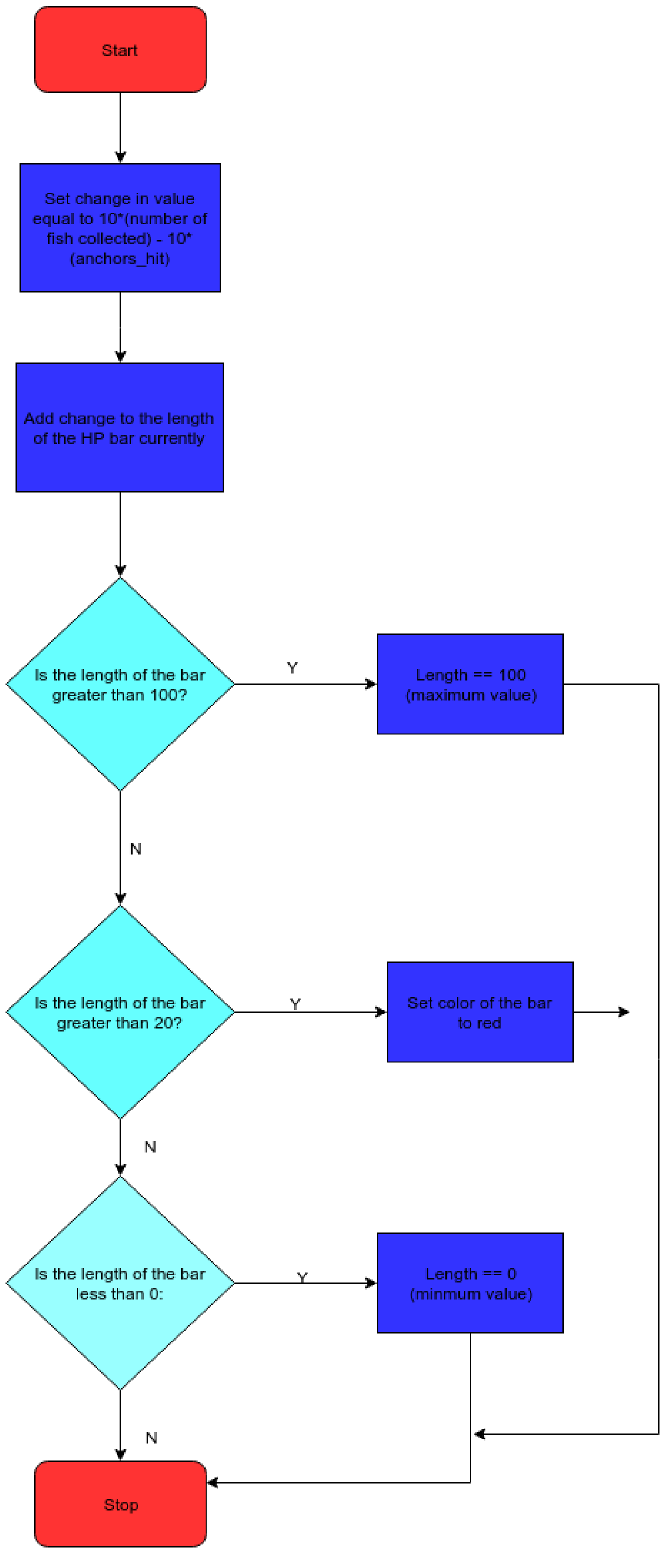
State Diagram

hp_level = HP_Level

color = GREEN
length = 60

The flowchart below shows the update function of the HP_Level.

HP_Level.update (anchors_hit, fish_collected)


Start
Set change in value equal to 10*(number of fish collected) - 10* (anchors_hit)
Add change to the length of the HP bar currently
Is the length of the bar greater than 100?
Y
Length == 100 (maximum value)
N
Is the length of the bar greater than 20?
Y
Set color of the bar to red
N
Is the length of the bar less than 0:
Y
Length == 0 (minmum value)
N
Stop

SpriteSheet (taken from ProgramArcadeGames.com, Chapter 13)

This class is used to make the process of using spritesheets more efficient. Rather than repeating the same lines of code every time you want to convert a sprite sheet, set a colorkey, and take a certain part of the image, this class allows you to input just the coordinates, width and height of the picture you want to take, and it returns an image that is ready for you to use in your code.

Class Diagram

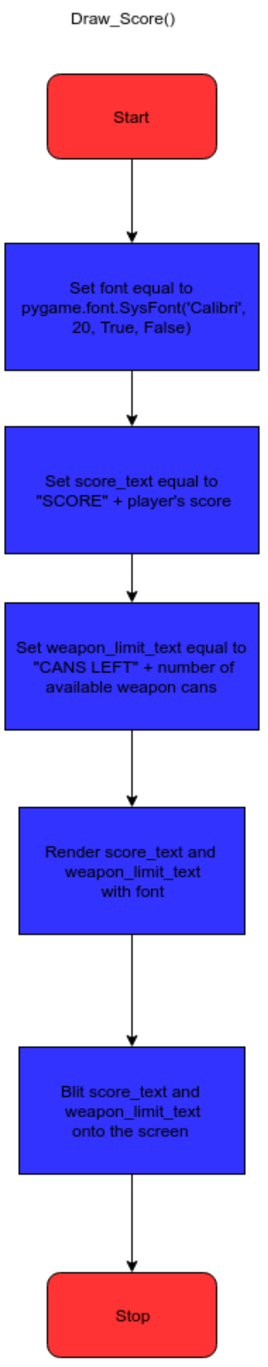
class SpriteSheet()
sprite_sheet: Object
get_image(x: int, y: int, width:int, height:int)

State Diagram

fish_image = SpriteSheet("fish.png")
sprite_sheet = "fish.png"

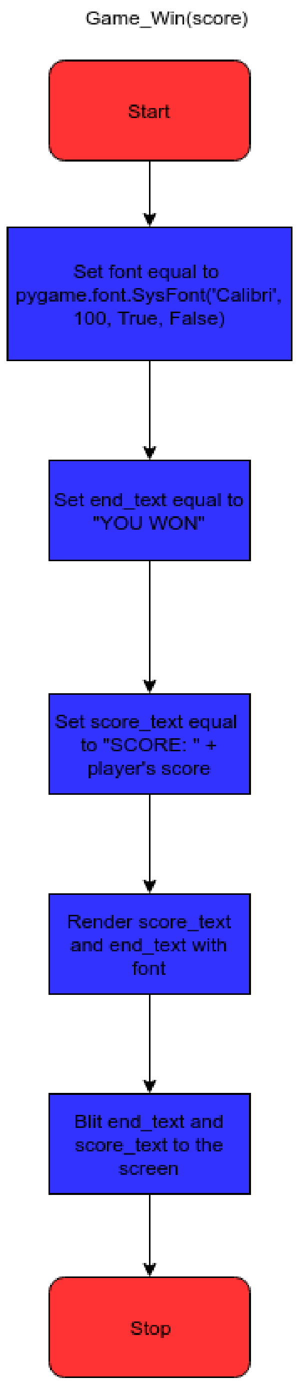
Draw_Score

This subprogram is used to display both the score and the amount of available weapon cans left to the user.


Draw_Score()
Start
Set font equal to pygame.font.SysFont('Calibri', 20, True, False)
Set score_text equal to "SCORE" + player's score
Set weapon_limit_text equal to "CANS LEFT" + number of available weapon cans
Render score_text and weapon_limit_text with font
Blit score_text and weapon_limit_text onto the screen
Stop

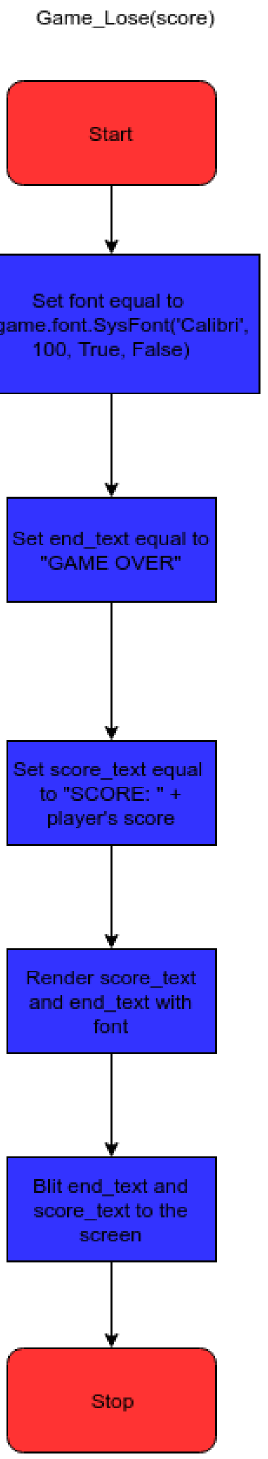
Game_Win

This subprogram is for the end of the game, when the player wins. It displays a final message as well as the score.


Game_Win(score)
Start
Set font equal to pygame.font.SysFont('Calibri', 100, True, False)
Set end_text equal to "YOU WON"
Set score_text equal to "SCORE: " + player's score
Render score_text and end_text with font
Blit end_text and score_text to the screen
Stop

Game_Lose

This subprogram will be initialized when the player runs out of HP. It displays the message “Game Over” as well as the player’s score.

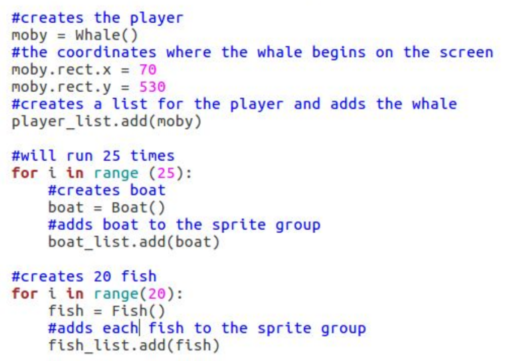

Game_Lose(score)
Start
Set font equal to pygame.font.SysFont('Calibri', 100, True, False)
Set end_text equal to "GAME OVER"
Set score_text equal to "SCORE: " + player's score
Render score_text and end_text with font
Blit end_text and score_text to the screen
Stop

Implementation Details and Deliverables

To implement all of these ideas into one game was not as difficult as I thought it would be. Using many of the concepts taught throughout the year, such as selection, sequence and repetition, paradigm programming and classes and objects, I was able to bring the game to life.

The player is made by creating an object of the class Whale. The player is added to its own sprite group.

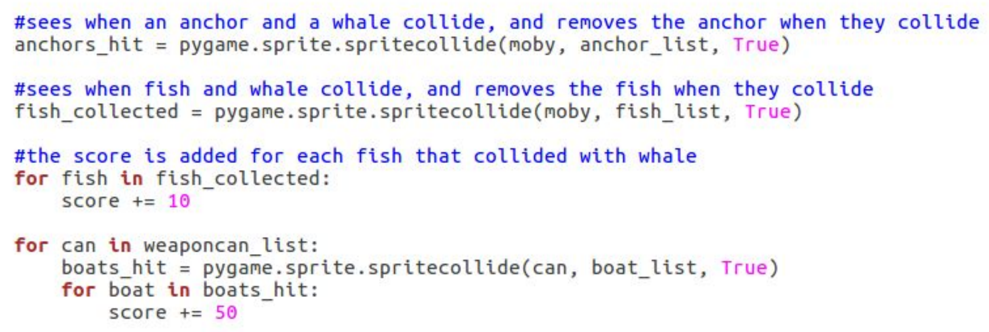
Using sequence, selection and repetition, the boats and fish are created.



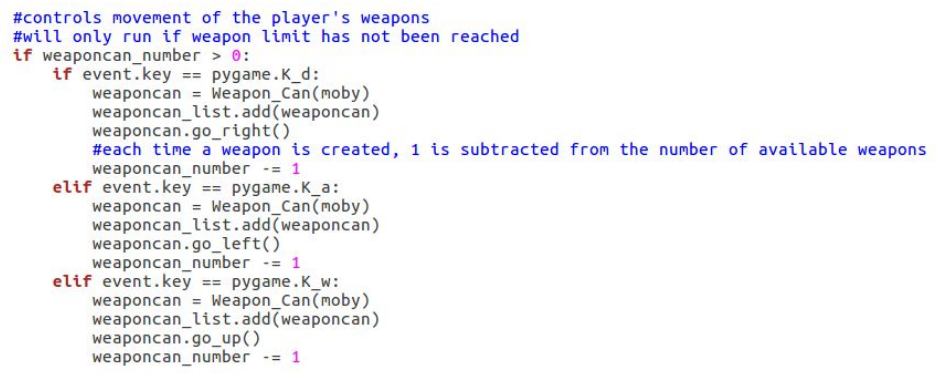
```
#creates the player
moby = Whale()
#the coordinates where the whale begins on the screen
moby.rect.x = 70
moby.rect.y = 530
#creates a list for the player and adds the whale
player_list.add(moby)

#will run 25 times
for i in range (25):
    #creates boat
    boat = Boat()
    #adds boat to the sprite group
    boat_list.add(boat)

#creates 20 fish
for i in range(20):
    fish = Fish()
    #adds each fish to the sprite group
    fish_list.add(fish)
```

The weapon cans are created only when the player presses a key out of A, D or S. An anchor is created only when the player goes near the whale.

Using the sprite collisions, all of the anchor hits, fish collections and boat kills were handled. Using variables outside of classes, the scores were calculated. This was a lot easier to manage the score rather than putting it inside of a class, since parameters and attributes would not be a hassle. (Note weaponcan_number and score in the screenshots below).

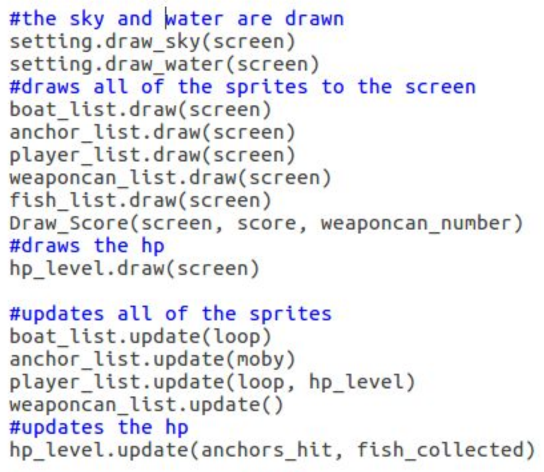


```
#sees when an anchor and a whale collide, and removes the anchor when they collide
anchors_hit = pygame.sprite.spritecollide(moby, anchor_list, True)

#sees when fish and whale collide, and removes the fish when they collide
fish_collected = pygame.sprite.spritecollide(moby, fish_list, True)

#the score is added for each fish that collided with whale
for fish in fish_collected:
    score += 10

for can in weaponcan_list:
    boats_hit = pygame.sprite.spritecollide(can, boat_list, True)
    for boat in boats_hit:
        score += 50
```



```
#controls movement of the player's weapons
#will only run if weapon limit has not been reached
if weaponcan_number > 0:
    if event.key == pygame.K_d:
        weaponcan = Weapon_Can(moby)
        weaponcan_list.add(weaponcan)
        weaponcan.go_right()
        #each time a weapon is created, 1 is subtracted from the number of available weapons
        weaponcan_number -= 1
    elif event.key == pygame.K_a:
        weaponcan = Weapon_Can(moby)
        weaponcan_list.add(weaponcan)
        weaponcan.go_left()
        weaponcan_number -= 1
    elif event.key == pygame.K_w:
        weaponcan = Weapon_Can(moby)
        weaponcan_list.add(weaponcan)
        weaponcan.go_up()
        weaponcan_number -= 1
```

Once the score, HP, collisions and movements were programmed, it was very easy to put the rest together, because all of the different groups could be drawn and updated with one command in the main program loop.



```
#the sky and water are drawn
setting.draw_sky(screen)
setting.draw_water(screen)
#draws all of the sprites to the screen
boat_list.draw(screen)
anchor_list.draw(screen)
player_list.draw(screen)
weaponcan_list.draw(screen)
fish_list.draw(screen)
Draw_Score(screen, score, weaponcan_number)
#draws the hp
hp_level.draw(screen)

#updates all of the sprites
boat_list.update(loop)
anchor_list.update(moby)
player_list.update(loop, hp_level)
weaponcan_list.update()
#updates the hp
hp_level.update(anchors_hit, fish_collected)
```

The balanced use of all of these different aspects made it very efficient and easy to create a game.

All of the files required to run the program are in the CPT folder. The evaluator must open the “CPTmoby.py” and run this module to run the game. If they wish to see any of the graphics or flowcharts in greater detail, all of the files with the corresponding names will be found in the CPT folder.

Testing

The game, after its creation, was tested for about two days by fellow peers. Lots of useful feedback was received and incorporated, mostly regarding the difficulty level of the game. Many reached the consensus that while it was very difficult in the beginning, it became very easy to beat once they were able to get used to the fast movements of the anchors. I decided to randomize the speeds of the anchors afterwards.

They also gave me more feedback on the controls, and many told me that due to the stiff movements, their game play was very restricted. This inspired me to create two new methods in the event that a key was released, so that the motion could remain fluid instead of being abruptly stopped, since most players pressed two different keys at once.

I also received some advice on the placement of the whale during the game. Originally, the whale was almost touching one side of the screen as the background scrolled. This made it difficult for the players to see approaching boats. A lot people suggested that I move the whale more towards the middle of the screen instead.

Using this feedback, I worked on and improved my game further.

Maintenance

The creation and implementation of “Moby’s Sick” has been a great learning experience. Since I really liked the theme and storyline of the game, I was motivated to work on the project and complete the game. I enjoyed that I could incorporate creativity into what I had learned over the course.

The aspect I am most proud of for this game are the graphics. Since I made all of the spritesheets myself, I was very happy with all of them, and they changed the whole atmosphere of the game. Somehow, having fun graphics made the game a lot more fun to play. I am also proud of all of the sprites and the collisions that I programmed. I worked really hard on figuring out how to implement the weapon cans and the anchors with the whale and the boats. It was so rewarding to see my work come to life on the screen.

The areas that need the most work are the organization of the code and the sprites. Near the end of the implementation period, I was struggling to finish and my coding became increasingly messier. If I had more time, I could clean it up and organize it a lot better. Right now, I think the main program and many of my classes are very messy and not as structured as they could be. Even though I had wanted my program to use complete abstraction, it did not turn out that way, which is something I can work on.

Another area that I need to work on are all of the sprites. I like the concept and storyline, but the game itself is very simple and repetitive. I would really like to work harder on the sprites so that they would be a lot more complex and have many more functions. I find that after a long time of playing the game, it is not as enjoyable and the game becomes very predictable and easy to beat. I would like new things to be introduced (such as upgrades, different weapons, costumes for the whale, new obstacles, etc.) throughout the game to make it more interesting.

Truthfully, all of the aspects of my game could use more time and work. The end sequence is very simple and basic, the screen was also very simple.

I received a lot of feedback from the students that have played my game. Generally, all of them enjoyed it and thought it was fun and interesting. I also received a lot of constructive criticism that helped me reflect on what I need to work on moving forwards.

Overall, I like the end result of the project and I enjoy the game I have created. I would like to continue to work on the game in the future and make further improvements.

References

Craven, Paul V. "13.4.5 Using Sprite Sheets." *Program Arcade Games*. N.p., n.d. Web. 2016 June 15. <http://programarcadegames.com/python_examples/en/sprite_sheets/>.