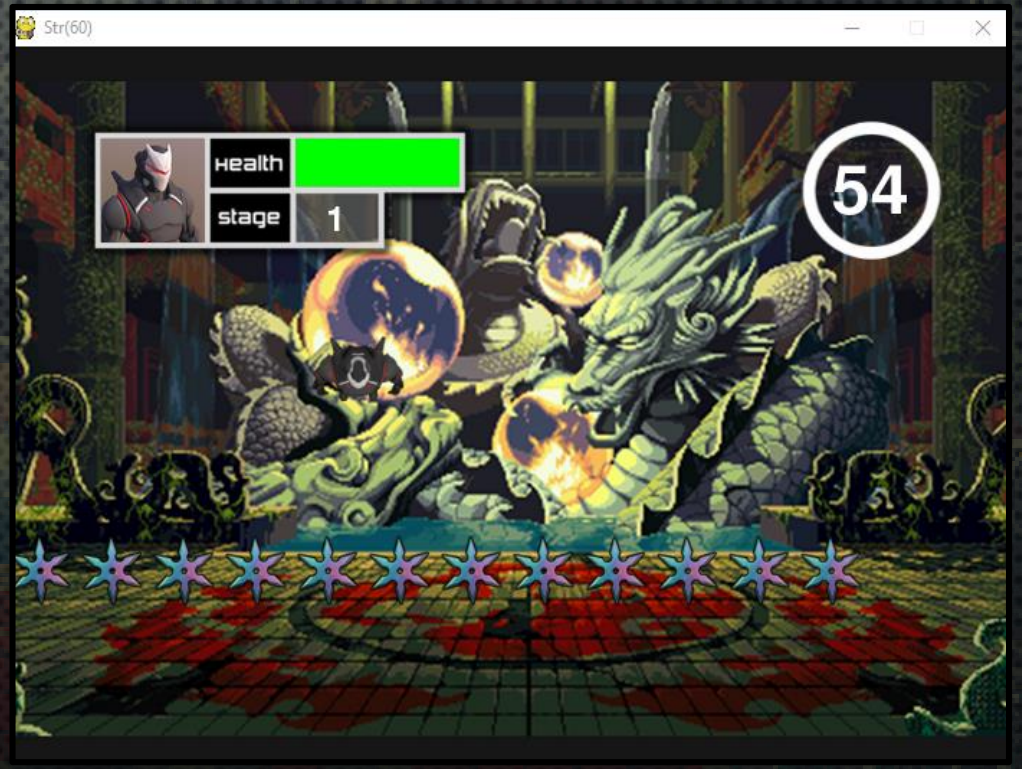

STR (60)
By, Ashhal S

WHAT IS IT

- Str(60) is a pygame video game in which you must dodge projectiles and answer multiple choice questions
- You must dodge the waves of ninja stars for 60 seconds.
- After every 20 second interval you must answer a computer science related question
- You begin the game with 3 lives; Hitting a projectile or answering a question incorrectly deducts 1 life
- To win the game you must pass all 3 stages without losing all of your lives



GAME FLOW

Lose Screen

Options Screen

Menu Screen

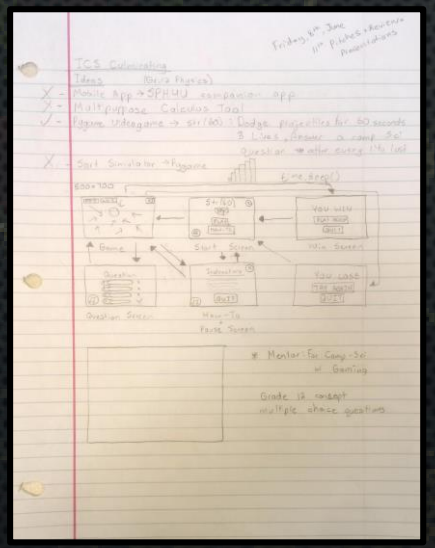
Play Screen

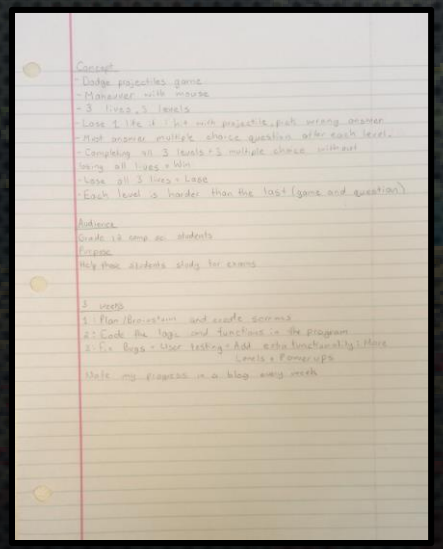
Question Screen

Win Screen

PLANNING

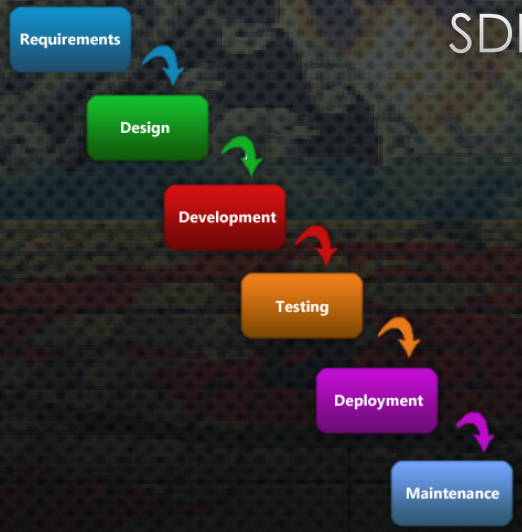
BRAINSTORM/CONCEPT

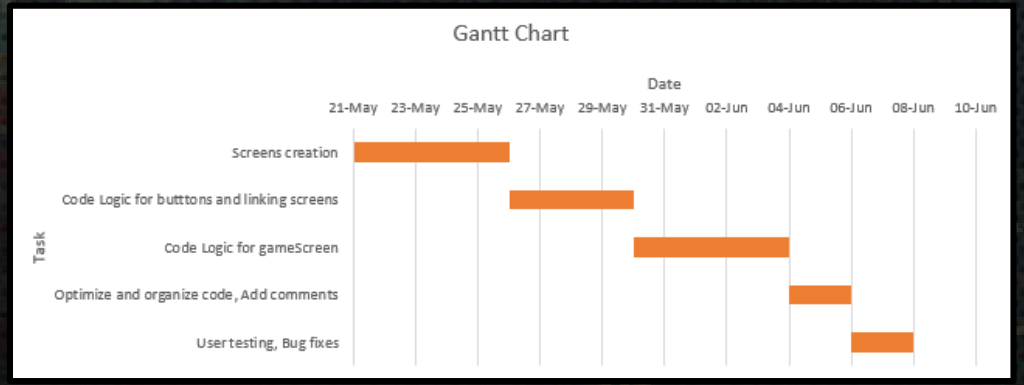




BLOG

SDLC & GANTT





PURPOSE

- I made this game to allow computer science students to study for exams in a fun, interactive way.
- Audience: grade 12 students

Which of the following is not a pillar of OOP?
Composition
Abstraction
Inheritance
Polymorphism

Which Big O Notation would be the most efficient in sorting?
O(1)
O(n)
O(log(n))
O(n^2)

The _____ of a variable dictates where it can be used
Scope
Stack
Function
Sentinel

What does Quick Sort use to break up and sort lists?
Pivots
Heaps
Ranges
Objects


THE CODE

VARIABLE AND CLASS DECLARATIONS

```python
#Imports the necessary libraries
import pygame,time,random

# Define some colors
BLACK    = (   0,   0,   0)
WHITE    = ( 255, 255, 255)
GREEN    = (   0, 255,   0)
RED      = ( 255,   0,   0)
YELLOW   = ( 255, 255,   0)

#Initiates pygame
pygame.init()

# Set the width and height of the screen [width, height]
size = (700, 500)
screen = pygame.display.set_mode(size)

#Adds title to window
pygame.display.set_caption("Str(60)")

#Loop until the user clicks the close button.
done = False

# Used to manage how fast the screen updates
clock = pygame.time.Clock()
```

```python
# -------- Main Program Loop -----------

class mySprite(pygame.sprite.Sprite):
    """
    Class used to create a usersprite.
    Each usersprite has an image, and a set amount of lives.
    May be used to create different player models for future iterations of the game
    """
    def __init__(self):
        self.image=pygame.image.load("UserSprite.png")
        self.rect=self.image.get_rect()
        self.lives=3


class projectile(pygame.sprite.Sprite):
    """
    Class used to create each projectile.
    Each projectile has an image, a location, and a direction
    """
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image=pygame.image.load("Projectile.png")
        self.rect = self.image.get_rect()
        self.rect.top=500
        self.rect.right=0
        self.direction=0,-7

#Game sound variable is defined and music is loaded
soundOff=False
pygame.mixer.music.load("SFV Concept AlbumSpectral Assassin Nash SFA RMX.mp3")
pygame.mixer.music.play(-1)

#A usersprite is created by inherting from the mySprite class
userSprite=mySprite()

#a list is created which will hold all the projectiles
allProjectiles = pygame.sprite.RenderPlain()
```

```
def getMultipleChoiceQuestions():
    """Function used to input the questions and correct answers for multiple choice questions from an input file"""
    #Initializes a dictionary
    questionsList={}

    #Reads data from a file
    myFile=open("Str(60)Questions.txt","r")
    allData=myFile.read()
    data=allData.split("\n")

    #Organizes data into the dictionary
    for i in range(0,len(data),5):
        questionsList.update({data[i]:data[i+1:i+5]})

    return questionsList
```

```
def evaluateAnswer(chosen,questionsList,randomQuestion,allChoices):
    """Function used to evalute whether the user has input the correct multiple choice answer"""
    #Blits either a correct or incorrect symbol beside the choices that are correct and incorrect.
    for i in range(4):
        if allChoices[i]==questionsList[randomQuestion][0]:
            correct=pygame.sprite.Sprite()
            correct.image=pygame.image.load("correct.png")
            screen.blit(correct.image,[575,180+(i*75)])
            pygame.display.update()
        else:
            incorrect=pygame.sprite.Sprite()
            incorrect.image=pygame.image.load("incorrect.png")
            screen.blit(incorrect.image,[575,180+(i*75)])
            pygame.display.update()

    #Resets the questions and answers and deletes the currect question from the possible question list
    time.sleep(3)
    if questionsList[randomQuestion][0]!=allChoices[chosen-1]:
        userSprite.lives-=1
    del questionsList[randomQuestion]
    randomQuestion="Blank"
    allChoices=[]
    return randomQuestion,allChoices,questionsList

#Creates a list which sets the difficulties of each stage.
difficulties=[5,3,2]

currentScreen="menuScreen"
```

EVENT HANDLING

```
while not done:
    # --- Main event loop
    for event in pygame.event.get(): # User did something
        if event.type == pygame.QUIT: # If user clicked close
            done = True # Flag that we are done so we exit this loop
        if currentScreen=="menuScreen":
            if event.type == pygame.MOUSEBUTTONDOWN:
            #Prevents user from selecting input while holding down mouse button
                if pygame.mouse.get_pressed() == (1,0,0):
                    mouseLocation=pygame.mouse.get_pos()
                    #If the user clicks the x button, the program will close
                    if mouseLocation[0]>=600 and mouseLocation[0]<=650 and mouseLocation[1]>=50 and mouseLocation[1]<=100:
                        done = True
                    #If the user clicks the music button, the music is toggled between paused and unpasued
                    elif mouseLocation[0]>=50 and mouseLocation[0]<=100 and mouseLocation[1]>=400 and mouseLocation[1]<=450:
                        if not soundOff:
                            pygame.mixer.music.pause()
                            soundOff=True
                        else:
                            pygame.mixer.music.unpause()
                            soundOff=False
                    #If the user clicks the play button the necessary variables are reset. This includes reseting lives, stage, time, projectiles, and choices
                    elif mouseLocation[0]>=200 and mouseLocation[0]<=500 and mouseLocation[1]>=200 and mouseLocation[1]<=350:
                        userSprite.lives=3
                        stage=1
                        timeLeft=60
                        startTime=time.strftime("%S")
                        questionStart=0
                        questionEnd=0
                        projectileTimerStart=time.strftime("%S")
                        allProjectiles.empty()
                        choice=0
                        randomQuestion="Blank"
                        currentScreen="playScreen"
                    #If the user clicks the options button, the user is taken to a screen that has instructions and credits
                    elif mouseLocation[0]>=250 and mouseLocation[0]<=450 and mouseLocation[1]>=375 and mouseLocation[1]<=450:
                        currentScreen="optionsScreen"
```

SCREEN MANAGEMENT

```
if currentScreen=="menuScreen":
    #If the user is on the menu screen, the correct background is blitted to the screen
    screenDisplyed=pygame.image.load("startScreen.png")
    screen.blit(screenDisplyed,[0,0])
```

```
if currentScreen=="playScreen":
    #If the user has 0 or less lives, the user loses
    if userSprite.lives<=0:
        currentScreen="loseScreen"
    #If the user passes stage 3, the user wins
    elif stage>3:
        currentScreen="winScreen"
    #If the user did not win or lose, the game logic continues
    else:
        #Blits the background to the game
        screenDisplyed=pygame.image.load("playScreen.png")
        screen.fill(BLACK)
        screen.blit(screenDisplyed,[0,0])

        #Makes the userSprite follow the mouse
        userSprite.rect.center=pygame.mouse.get_pos()
        time.sleep(0.0001)
        screen.blit(userSprite.image,userSprite.rect)

        #Sets the difficulty for the current stage
        difficulty=difficulties[stage-1]
```

```
def spawnWave(size,allProjectiles):
    """Function used to spawn a wave from below"""
    hole=random.randrange(0,650,50)
    for i in range(size[0]/50):
        currentProjectile=projectile()
        currentProjectile.rect.left=i*50
        if currentProjectile.rect.left==hole or currentProjectile.rect.left==hole+50:
            currentProjectile.rect.right=0
        allProjectiles.add(currentProjectile)
    return allProjectiles

#Allows the wave to spawn every 5 seconds
projectileTimerEnd=time.strftime("%S")
if int(projectileTimerEnd)<int(projectileTimerStart):
    projectileTimerEnd=int(projectileTimerEnd)+60
if int(projectileTimerEnd)-int(projectileTimerStart)==difficulty:
    spawnWave(size,allProjectiles)
    projectileTimerStart=projectileTimerEnd

#Checks whether the user has collided with a projectile, if not the wave is moved up
for item in allProjectiles:
    if (pygame.sprite.collide_rect(userSprite, item)):
        userSprite.lives-=1
        item.rect.topright=(0,500)
    if item.rect.right!=0:
        item.rect=item.rect.move(item.direction)

#Draws the projectiles to the screen
allProjectiles.draw(screen)
```

```
#Draws the playerBar to the screen
playerBar=pygame.sprite.Sprite()
playerBar.image=pygame.image.load("playerBar.png")
screen.blit(playerBar.image,[50,50])

#Draws the timer border to the screen
timer=pygame.sprite.Sprite()
timer.image=pygame.image.load("timer.png")
screen.blit(timer.image,[550,50])

#Allows the user to have unlimited time to answer multiple choice questions
questionTime=questionEnd-questionStart
timeLeft+=questionTime
timeLeft=int(timeLeft)

#Determines the time left in the game
endTimer=time.strftime("%S")
if endTimer<startTime:
    timeLeft-=60
timeLeft-=(int(endTimer)-int(startTime))
startTime=endTimer

#Blits the time left inside the timer border
myFont=pygame.font.Font(None,70)
text=myFont.render(str(timeLeft),True, WHITE)
if len(str(timeLeft))==2:
    screen.blit(text,[572,77])
elif len(str(timeLeft))==1:
    screen.blit(text,[585,77])

#Launches a multiple choice question after 20 second intervals
if timeLeft in [40,20,0]:
    currentScreen="questionScreen"
    allProjectiles.empty()
    questionStart=time.time()
else:
    questionStart=0
    questionEnd=0
```

```
#Draws the player logo to the screen
playerLogo=pygame.sprite.Sprite()
playerLogo.image=pygame.image.load("playerLogo.png")
screen.blit(playerLogo.image,[64,64])

def playerStatus(stage):
    """Function used to calculate and blit the player's current lives"""
    #Sets the lives colour to red, yellow, or green if the user has 1, 2 or 3 lives respectfully
    livesColour=[RED,YELLOW,GREEN]
    livesColour=livesColour[userSprite.lives-1]

    pygame.draw.rect(screen,livesColour,[199,64,38*userSprite.lives,33])

    myFont=pygame.font.Font(None,35)
    text=myFont.render(str(stage),True,WHITE)
    screen.blit(text,[220,108])

#Calls the function that blits the user's lives
playerStatus(stage)
```

BETA TESTING AND CONCLUSION

USER TESTING

- THE GAME WAS TESTED BY INDIVIDUALS FROM INSIDE AND OUTSIDE THIS CLASS
- ALL BUGS WERE FIXED AND SUGGESTIONS WHERE ACCOUNTED FOR (HITBOXES)
- THE GAME DATA IS AVAILABLE FROM MY WEBSITE (HTTPS://ASHHALSYED.WIXSITE.COM/CULMINATINGPROGRESS)

Tester Initials (A.S)	General Thoughts (Good, Bad, Why)	Bugs / Glitches	Suggestions for improvements
J.S	• Good design	• No bugs	• Increase sensitivity
H. G	• Layout is well designed	• Nothing happened when I lost all my lives	• Fix bug • Make last level easier
A.B	• Very nice GUI	• Hitbox issues	• Increase framerate if possible
AS	• Graphics are very nice, easy to use	• Hitboxes are slightly off	• Fix hitboxes if possible
A.M	• Smooth Animation and good integration of music and visuals	• Hitboxes were slightly larger than object	• Fix hitboxes, more stages and different types of projectiles