

Project Proposal

Project Description

My term project is called 'League of Legends 1v1.' This project will be a simplified version of League of Legends, which is a 5v5 multiplayer online battle arena video game. The objective is to destroy the opponent's Nexus, or base, before the opponent does after destroying towers. The main interactions between the user and the project are that players have to move around with right click mouse button and auto-attack or use skills to kill the wave of minions and the enemy champion.

Competitive Analysis

Because this project is the remake of an existing online game, it will borrow the main concepts of the game. For example, the main objective will be similar in that whoever destroys the base wins the game. The idea of towers, minions, and skills will be used too. There will be a wave of 3 melee minions and 3 ranged minions coming from each side of the base. A tower will hit the minion or the opponent champion within its range. The player uses auto-attack or skills to kill the minions or kill the enemy AI. The project is specifically adapted from a game mode ARAM from League of Legends, in which unlike the normal game mode, there will be only be one path from nexus to nexus. Hence, the map in the project will be adapted from the map of ARAM mode.

However, the project is a simplified version of the actual game, so some features will be simplified or removed. As an example, in the project, there will be only one champion to select from, the skills will be altered to make codes easier, and the graphics will be closer to 2D. Unlike the actual ARAM mode, which is a 5v5 game, this project will only have 1v1 format, where a player is playing against an AI.

Structural Plan

The project will use pygame as the module and implement OOP for each group of objects. The code will consist of minion class, item class, player class, tower class, Nexus class, and an AI class, so that each object can perform different attributes. In the main loop, the instances will be called to run the program with time. For organization, all the drawing instances will be called in the redrawGameWindow function, which will then be called in the main loop. In the folder, there will be another folder, where all the images will be stored and called, to avoid confusion between storyboard, project proposal, images, and the python file.

Algorithmic Plan

The trickiest part of the project is coding the AI player. The objective of the AI is that it should try its best to kill the player and destroy the towers and nexus as soon as possible. Hence, the movement of the AI will be adjusted depending on the player's health and minion's health. If the player's health is low, the AI will be aggressive by moving towards and attacking the player. If the situation is the opposite, the AI will be passive and go back to the base to regain health. The AI will push the wave as fast as possible by auto-attacking and using skills on the minions. When the player is at the base or dead, the AI will hit the minions as fast as it can, and attack the turret. The key feature of the project is the movement of the AI depending on the movement of minions and the champion.

Timeline Plan

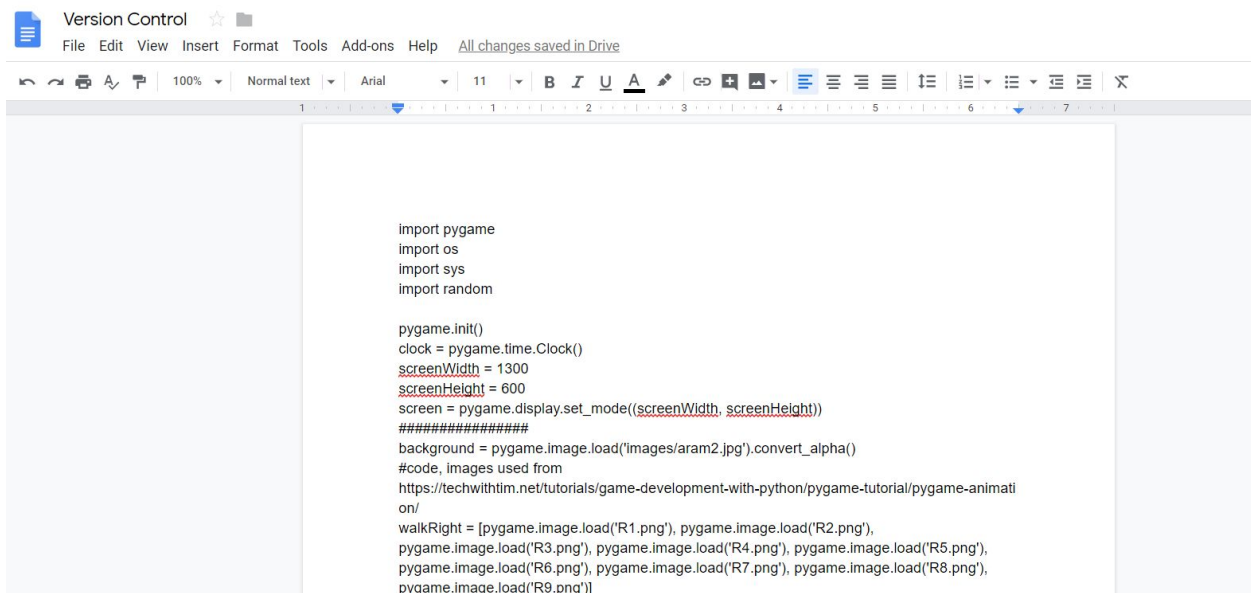
I will finish fixing the error of scrolling by Wednesday April 17th. I will finish collecting and assigning sprites for the champions, minions, towers, turrets, and items until Thursday April 18th. I will finish setting the boundaries of the map by Thursday April 19th. I will finish coding the auto-attacks of the minion and the champion by Friday April 20th. I will finish making the user and minions' health bar by Saturday April 20th. I will finish making the skills and cooldowns of the champion by Sunday April 21st. I will finish creating the algorithms for the towers and nexus by Sunday April 21st. I will finish making the algorithms for the AI by Tuesday April 23rd. If I have time, I will finish making the items and gold system by Wednesday April 24th.

Module List

The only module that will be used for the project is Pygame.

Version Control Plan

I will back up the code by copying and pasting the code into a google doc provided by the University like in the picture below. Then, if my computer malfunctions, for example, I can simply go onto google doc from a different computer and easily access the code.



```
import pygame
import os
import sys
import random

pygame.init()
clock = pygame.time.Clock()
screenWidth = 1300
screenHeight = 600
screen = pygame.display.set_mode((screenWidth, screenHeight))
#####
background = pygame.image.load('Images/aram2.jpg').convert_alpha()
#code, images used from
https://techwithtim.net/tutorials/game-development-with-python/pygame-tutorial/pygame-animati
on/
walkRight = [pygame.image.load('R1.png'), pygame.image.load('R2.png'),
pygame.image.load('R3.png'), pygame.image.load('R4.png'), pygame.image.load('R5.png'),
pygame.image.load('R6.png'), pygame.image.load('R7.png'), pygame.image.load('R8.png'),
pygame.image.load('R9.png')]
```

Because google doc does not account for the indentation, I will also save my code in the text file, which allows proper use of whitespace.

version control - Notepad

File Edit Format View Help

```
import pygame
import os
import sys
import random

pygame.init()
clock = pygame.time.Clock()
screenWidth = 1300
screenHeight = 600
screen = pygame.display.set_mode((screenWidth, screenHeight))
#####
background = pygame.image.load('images/aram2.jpg').convert_alpha()
#code, images used from https://techwithtim.net/tutorials/game-development-with-python/pygame-tutorial/pygame-animation/
walkRight = [pygame.image.load('R1.png'), pygame.image.load('R2.png'), pygame.image.load('R3.png'), pygame.image.load('R4.png'), pygame.image.load('R5.png'), pygame.image.load('R6.png')]
walkLeft = [pygame.image.load('L1.png'), pygame.image.load('L2.png'), pygame.image.load('L3.png'), pygame.image.load('L4.png'), pygame.image.load('L5.png'), pygame.image.load('L6.png')]

class Background(pygame.sprite.Sprite):
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def draw(self, screen):
        background = pygame.image.load('images/aram2.jpg').convert_alpha()
        background = pygame.transform.scale2x(background)
        screen.blit(background, (self.x, self.y))

class Player(pygame.sprite.Sprite):
    def __init__(self, x, y, width, height):
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.speed = 10
        self.walkCount = 0
        self.left = False
        self.right = True

    def draw(self, screen):
        if self.walkCount + 1 >= 27:
            self.walkCount = 0
```