Project Proposal

Project Description:

- Game title: Live by Daylight
- Live by Daylight is a one-player top-down view 2d game that the player needs to control P1 to escape from the gate by opening one of the chests on the map. During the gameplay, P2 —- the AI attendant will collaborate with the player by opening chests or kiting the killer. Meanwhile, the AI killer will wander around different areas on the map to find the survivors. After being attacked twice, the survivor will be brought to the jail. And if the survivor is brought to the jail twice, it dies. The game ends if the player escaped or died, but continues even if P2 dies.

Competitive Analysis:

The core algorithm of this project is the code of the game AI for the killer. Consider that the information of the survivor is available for the killer, A* search projects will be similar to my intended code for finding the closest path from the killer to the survivor. However, a condition of the distance between the killer and the survivor is added, so the survivor will only be "discovered" and chased by the killer within a certain distance. When not chasing any target, the killer will wander around areas on the map. Also, the shortest distance will be constantly changing due to character motion, thus the shortest route will be updated every time interval.

This project is adapted from the horror game Dead by Daylight, so the game structure is very similar to it. However, the scale is slightly reduced to a single-player game with an AI enemy. The player will be the survivor that aims to escape from the map, and the killer will stop the survivor from escaping by attacking them. After two attacks, the survivors will be brought to the jail. If a survivor enters the jail twice, it dies. As the survivor being controlled by the player dies, the game is over. To open the gate, the survivors have to open one of the two chests on the map. Then, if the player arrives at the gate, the player wins and the game ends.

Structural Plan:

- The processes of the chests and the game status are constantly tracked in the timer fired function, and the main app message is updated correspondingly. The A* algorithm that calculates the closest path to the target is triggered every time interval to update the path and direct the killer's motions. The survivor AI is also updated every time interval in the timerFired function.
- Key presses are used to control the movement of survivor A as well as actions such as opening the chest (and of course the cheat keys :)).
- The obstacles including the walls in the map are organized in the class of obstacles and the subclass of passable obstacles such as the gate and the jail. The characters belong to either the killer or survivor subclass of the character class that records their status, data, and creates character stripe animations.

Algorithmic Plan:

The most difficult part of my project is the game AIs which include the killer AI and survivor AI:

Killer

- When there is no survivor in a certain radius around the killer, the killer goes around the court clockwise to search for survivors.
- When a target has been discovered by the enemy, A* search will be used to find the closest path from the killer to the survivor and direct the movement of the killer.
    - The origin point is first set at the location of the killer, while the endpoint is set at the location of the target survivor and updated every time interval. The origin point is the first "current node" being investigated.

        - From the current node, find every valid child node and calculate its f value, which combines the certain distance from the origin and the estimated distance from the child to the endpoint. Then, this child node is added to the list of nodes that will be later investigated (yet to visit list).
        - Then find the next current point based on the low f value and the generation of the node. This means the node with the lowest f value in the yet to visit list will be the next “current node” to be investigated because a small f value is preferred (indicate short distance), and usually small generation leads to small f so the default current node is the 0th node in the list (generally sorted by generation due to the mechanism of appending).
        - If the current node is already the endpoint, the data of this node will be translated to a path by retrieving backward through the parent data stored as one of the attributes of the node object. The path will later be used to move the killer through the grids.
    - While bringing a dying survivor to the jail after two attacks, the killer will ignore the other survivor.

Survivor

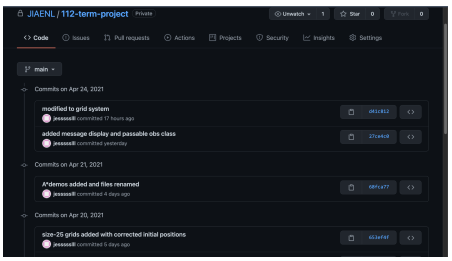
- When the killer is not in a certain radius around the survivor AI, it will approach the closest chest to start opening the chest. (the path will be found using the same A* algorithm described above)
- If the killer is in fact within the radius of detection, the survivor AI starts to escape by the following logic.
    - The opposite direction from the survivor to the killer will first be calculated and used as the default direction to escape.
    - If the direction eventually leads to a blind alley on the terrain, the angle of the direction vector will either start to increase or decrease by random selection in order to detect a possible route.

Timeline Plan:

- 4/26: complete chest opening mechanism and game over detection
- 4/28: complete enemy default route
- 4/29: complete enemy algorithm including path search and attack detection
- 5/1: complete jail mechanism
- 5/3: complete survivor B algorithm

Version Control Plan

- The code and files of this project will be constantly backed up on GitHub whenever a new feature is added or a major change in code is made.



Module List:

- As the main part of this project based on A* algorithm, no extra modules are planned to be used. But there is still a possibility that PyAudio will be used in later stages just to add some background music.

TP2 Update

- None

TP3 Update

- Update in survivor B AI
    - If the killer is in fact within the radius of detection, the survivor AI starts to escape by the following logic according to the difference in their row, column, and the available space around it.
        - If the difference between rows or columns is greater than the other, it is prioritized in the next step as the ideal position is to keep the killer vertically or horizontally away from the survivor so the killer AI couldn't cut corners. If there is an obstacle in this default direction, the second choice is the other component of the vector.
        - However, choosing the other component rises further issues. If the other component is zero, the AI will randomly choose from the available moves in that direction. (i.e. if the row direction is unavailable but the column component of the relative position is zero, the AI will either go left or right if both ways are plausible.
        - For the case of corners where the directions of both components are not available, the survivor will choose the opposite direction of the shorter component in order to leave the corner in the shortest time and achieve the greatest distance from the killer.
        - To avoid endless chasing, the design of the survivor AI creates a slight pause between every move so that the killer will be able to reach the same cell as survivor B. If so, the survivor AI will randomly move away from the killer and recalculate as the above processes describe. In fact, the actions are recalculated after every step of the survivor AI, which is the main contributor to the slight pause.
    - The initial position is randomly generated
- Update in Killer AI
    - The detect range is not constant anymore. To cover the need of chasing detours due to the obstacles, the detect range increases after detecting a target and reduces after losing it.
- Update in the display: only the area around survivor A is visible. But if the game starts to run slowly, the user could turn the lights on by hitting the 'L' key. Additional cheat keys of 'B' and 'K' that freezes survivor B and the killer were added.