Evan Schipellite and Jak Tiano

Time-Tanks Technical Design Document

Prepared by: Evan Schipellite

Required Components

- An alternative graphics library linked correctly to the project base
    - This will likely be SDL 1.2.
    - Project components relating to Allegro should be swapped with SDL features.
    - The project itself should run fairly well with the changes and should feature SDL aspects to manage the buffers, drawing, rotating, text, primitives, and events.
- A partial Player class
    - A player class containing information such as the player’s location and rotation should be created.
    - This class may have further developed features, but the majority of the player-related aspects may be maintained by a type of HUD class that keeps track of level data relating to the player’s score, attempted tries left, and other aspects.
    - Therefore, the player class should primarily be concerned with managing the drawing of the player object, player turret, and possibly collision checks through events.
    - Upon player death / end time, the player’s timestamps should be sent to the ghost manager to create a player-based ghost object.
    - This class should also keep track of when the player has fired and when the player can fire again. This will allow for a delay to prevent the player from rapidly firing. By firing, an event should be dispatched to launch a projectile at a given location.
    - This class should also keep track of the Player’s health in the game, which may decrease from enemy projectile collisions, as well as a timer.
    - The Player class will save information relating to the player’s actions, and then utilize an event to send out the information upon death / end of time.
- Enemy Manager
    - An enemy manager should allow for a certain number of enemies to be spawned at specified locations. The game should keep track of these enemies placed in the world, and upon being destroyed in the game, they should be removed from the manager.
        - Upon being destroyed, an event should be created to coordinate with the Ghost Enemy Manager in order to monitor which enemies should become part of the replay feature.
- Enemy Class
    - The enemy class should keep track of the enemy’s location, rotation, and current state. Upon being destroyed in the game, the level manager should allow for the enemy to be deleted, and an event should be dispatched to create an enemy ghost based on time stamps.
    - It is likely that the enemy’s update / movement procedure will also contain simple AI aspects in order to detect a nearby player and then begin moving / targeting the player. Upon going out of range the enemy should proceed to move to the player’s last known location.
    - If no player is present, the enemy should move randomly in an idle motion around a small area, simply detecting collisions with walls and moving small amounts in random directions.

    - The Ghost class will likely save information based on its own actions, sending the information out upon being destroyed.
- Ghost Manager
    - It may be the case that the Manager may be divided into two separate systems where the enemy and player ghosts are kept track in separate manners. However, ideally it would make the most sense for the recording process to be determined in such a similar manner that the Ghost Manager will simply take in the time stamps, object, locations, and rotations in order to recreate the scenario.
    - Therefore, the Ghost Manager will be able to listen for events to create the entities based on time stamps, objects, locations, and rotations. The Ghost Manager will then create either a Ghost enemy or Ghost player and then save the information in the Ghost Manager. Whenever the game begins again, meaning the player had died / run out of time, the Ghost will be added to the visuals and then will act out following its information to imitate its previous actions in the game.
    - If a ghost reaches the end of its information sets, meaning the player had survived longer than the Ghost was around for, the Ghost should simply vanish.
- Ghost Class
    - The Ghost class will contain the information regarding the time stamps, the locations, the rotations, and the Ghost visual. It will be primarily managed by the Ghost Manager in terms of how it is run through, but all of the information will be stored in the Ghost itself.
- Input Manager
    - The Input Manager should simply listen for basic key buttons such as 'ESCAPE' in order to quit the game, as well as 'SPACE' and other keys to navigate through the menus. Gameplay will likely be completed through keyboard features. It is likely that the player will rotate the turret utilizing the A and D Keys, Fire using the SPACE Key, and move with the Arrow Keys.
- Projectile Manager
    - A Projectile Manager should simply update and monitor all of the projectiles in the game world. Upon being created by an event, a Projectile will be added to the Manager and will be deleted upon a located collision.
    - Depending on the type of collision, an event to reduce the health of an Enemy or Player will be sent out. Regardless, the Projectile should be destroyed.
- Projectile Class
    - The Projectile Class will keep track of the velocity, location, angle, and visual of the Projectile being fired. It should also note the association (Player or Enemy) in order to monitor which objects it can successfully collide with over the course of the game.
- Level Manager
    - The Level Manager should simply monitor the amount of Levels in the game.
- Level Class
    - The Level Class should concern itself with the world visual, bounding boxes, level difficult, player data, and tank data.
    - It is likely that all information required for the HUD and Player might be based in the Level to obtain from a Level Data system at the start of the game when all of the levels are created.
    - Therefore, information will be passed to the HUD and Player when the Level becomes active in the Level Manager.
    - It may be possible for each level to own its own start / end screen that will introduce the player with win / loss screens over the course of gameplay.

- Menu Manager
    - The Menu Manager will contain all of the information relating to the start screen, language screen, and option menu.
    - Depending on the given menu presented, the Menu Manager should allow the game to listen for certain prompts and then send out events accordingly in order to adjust the game's features.
    - Adjusting the difficulty, language, or sound, the game should send out events to adjust these features.
    - Localization can likely be done through text files that are loaded depending on the language choice made in the options.
- Saved Game Class
    - This class will likely hold the information relating to the current gameplay upon exiting the game. Upon starting the game, one of the options on the start screen will likely allow the user to initiated this class to load the saved information and continue the game.
- Event Information
    - A number of undetermined events will likely be created in order to pass information concerning the adjustment of player health, gameplay features, and enemy health.
    - The most notable event component will likely relate to the creation of an event that will assist in determining the Ghost information. It is likely that an event containing time stamps, locations, rotations, and a visual type could be utilize for the replay feature of the game.
    - This Replay Feature will likely be the source of the external research component of the project.
- Sound Manager
    - This system will store and manage all of the sound components in the game. Upon being requested by an event, the Sound Manager will play a certain sound or stop sounds depending on the game state.
- Sound Class
    - The sound class will contain information relating to the current sound clip to play, playing the information at the request of the Sound Manager.
- HUD Class
    - The HUD class will store and present the various pieces of information relating to the gameplay.
    - The Player health, tries left, level, and potential score may be displayed on the HUD.
    - The HUD also may present the FPS of the game in the area.
- Bounding Box
    - This class should create a simply invisible bounding area to represent a wall. Basically this will allow for the objects to have bounding boxes, including the Player, Enemy, Projectile, and Walls.
- Wall Manager
    - A Wall Manager should simply keep track of the collision walls in a given level. Upon interaction with any object, it should prevent movement through it.
- Wall Class
    - The Wall class should essentially just contain a bounding box, as well as the ability to be detectable as a wall during collisions.

Risk Assessments

- **High:** Evidently, creating the replay feature may prove to be the most difficult task of this project, largely because it is an unknown component that has not been done before by the programmers. Therefore, it's actual creation may diverge from the original assumptions due to the more efficient methods that may be observed through research.
- **Medium:** Creating the Sound Manager and Sound System will take some time because it is also an unknown feature. It may not be incredibly difficult, but locating proper sound files and adding them to the game in an appropriate fashion may take some planning.
- **Medium:** Detecting Collisions utilizing the bounding boxes may take some time to format in a universal manner, but this is simply due to the fact that there is no current collision method available in the project base.
- **Low:** Localization may take time to properly set up, but it is likely that it will simply be a matter of formatting a few text files in order to read them in depending on the given language selected.
- **Medium:** The saved game class may take some time to properly create. Although saving information is simple enough, saving the information required to effectively imitate the game state may require some extensive testing.
- **Medium:** Evidently, working with SDL may prove troublesome, not only in terms of linking, but also in terms of future development. Running multiple graphics could potentially cause later problems, although those issues may simply need to be prioritized upon appearance.
- **Medium:** Creating Ghosts may require some time to set up the system, simply due to the fact that saving all of the required information may not be as simply as originally intended. It may require more research and knowledge of generating and relating time-stamps to properly set-up all of the Ghost movements.