Asteroids

Technical Design Doc 1.0

By Jonelle Lawler & David Whiteford

Overview

This 2D space shooter will have 7 different classes. All of these classes will have their own functions related to what the class is for. This will include functions for movement, setting up the sprites, collisions, menus for the game and others that will help the game function.

Classes

Game

Game events are written in the game class. Some of the functions that will be contained within this class include:

- **Game**
  This is where the game window is set
- **Update**
  The move function will be called here
- **Run**
  This is where the frame updates are set
- **Draw**
  This function is used to draw different sprites to the screen.
- **loadContent**
  This is where sprites for the game are loaded.
- **Follow**
  This function is used for the follower enemies. As it is called in the Game.cpp class it must be passed the player object by reference. A for loop will be used here and the if statement within it will be checking if the enemy in the array is alive. If it is, the follow function will be called and set to the current enemy in the array.

Menu

- **Main Menu**
  Text will be loaded in the menu class, displaying all its sections.
  We will use the key press checking function here to check if the right mouse button has been clicked on any particular section.
  This will navigate the player to the next menu which can be the help menu, map menu, or the upgrade menu.
- **Help Screen**
  Text is outputted to the screen here. The player can scroll through the menu using the mouse and exit using the Esc key, so the key press checking function will be used here.

- **Map**
  The map will be made up of different planets the user can play on. Information on the amounts and different types of resources on each planet will be provided if the player hovers over one; This will be checked by dividing the mouse coordinates by the object size. If a user right clicks on a planet, a bool will be set to true and the game level will be loaded under that condition. Some levels must be unlocked by earning more points. This will be checked with a loop - example: if the number of points is between 5 and 10, level will load after being selected.

Hanger

- **Sprite**
  Allows for the loading of any sprites that will be used in the hangar menu.

- **Upgrade**
  All of the upgrades will be activated by a boolean and an if statement that will check if the boolean is true to activate. This will activate once the player has purchased any upgrades related to the boolean.
    - The hanger will allow the player to upgrade their ship. There will be several modules that the player will be able to add to their ships. These include health upgrades, fuel upgrades and shield upgrades that will be able to be added to the ship.
    - The player will select the module they wish to upgrade and then it will display all the upgrade options available. This will be done with a mouse click. The game call another function from the upgrade function for the part they wish to upgrade.
- **Fuel**
  This function will allow the player to decrease the amount of time it takes the player to run out of fuel. It will also display the amount of resources and scrap metal it cost the player to upgrade. This function will be called from the upgrade function. To escape the function and return to the upgrade function the player can press Esc key.

- **Health**
    - This function will also be called from the upgrade function and will allow the player to increase the amount of health they have.
    - This function will display the amount of resources, credits and scrap they have, which will be done with both sprites and text
    - They can leave the function by pressing the Esc key, so key press checking will be used here.

- **Shields**
    - This function will also be called from the upgrade function and will allow the player to increase the amount of shields they have.
    - This function will display the amount of resources, credits and scrap they have.
    - They can leave the function by pressing the Esc key (key press checking).

- **Shooting**
  This function will also be called from the upgrade function and will allow the player to upgrade will increase the speed the bullets and in turn increase the rate of fire of the bullets. This function will display the amount of resources, credits and scrap they have. They can leave the function by pressing the Esc key.

Player

- **Move**
    - Keyboard press checking used to move the player. Player's x and y positions altered accordingly.
- **Shoot**
    - Keyboard press checking with a boolean here. If the key is pressed, the bullet will be drawn from the player and move in the direction set in the direction function.
- **setPosition(float t_xPos, float t_yPos)**
    - This function accepts two arguments: the x position and the y position of the player and sets that as their location
- **setTexture**
    - We will set the default player sprite here (the starter sprite, before the player has moved)
- **getBody**
    - Returns the body of the player and can be used to move the player and used for collisions that the player is involved in.
- **getDirection**
    - Direction is gotten from what key the player has pressed (up, down arrow etc.) and used to set where their bullet goes and to rotate the sprite
- **getShield**// Controls shield power up
    - Returns the value of the shields
- **setLives**
    - Number of lives the player has is set here. Default = three. One taken every time player takes damage (collision detection with bullets and aliens)

- **shield**
    - A shield is activated if a player takes damage. A boolean is set to true when this occurs. This shield will be active for 30 seconds. The timer bar located on the top right corner of the game window will decrease to show the player how long they have left - this will be done by subtracting from its x coordinate in a loop. For example: while the boolean is equal to true, subtracting this amount from the x coordinate. When the x coordinate reaches 0, the boolean will be set to false.

Asteroids

- **Move**
    - This controls the movement of the asteroids. They will move in random directions (within a set range) by altering their x and y coordinates, adding a speed to them.
- **Explosion**
    - A bool will check for collision detection between player bullet and an asteroid.
    - Collision detection will work by

Basic Aliens

- **Set Texture**
    - A default texture will be set as all aliens start the game facing south.
- **Move**
    - The basic aliens will move in a random direction (within a set range).
    - This will be done by altering their x and y coordinates. We will do this by adding their speed to them.
    - The direction of the alien will be set when they move - for example, if we are subtracting from the y coordinate, the alien is going to face East.
    - The default sprite set in the function above will change accordingly when the alien moves. This will be dictated by the direction they are facing. For example, if the enemy is moving from upwards from their starting position (facing south), we will rotate the sprite 180° using the .setRotation(180) function.
- **Shoot**
    - These aliens can shoot at the player. The function will get their direction and a bool, when set to true, will allow the projectile to be drawn. It will move by adding speed to its x or y coordinate; Which one depends entirely on the direction, which will be calculated by the move function and called in here.
- **Set Position**

    - The starting position of the alien will be randomly generated, again, within a range.
- **Set Texture**
    - A default texture will be set as all aliens start the game facing south.

Follower Aliens

- **setPosition**
    - Sets the start position of the alien.
- **setTexture**
    - Sets up the aliens' image.
- **Follow(Player &aPlayer)**
    - Controls the movement of the aliens.
    - The player object is passed by reference.
    - This function will get the alien's current position.
    - Get the player's current position.
    - Compare their positions and if the x position of the alien is less than that of the player's, increment it.
    - If the alien's x position is greater, it will decrement it.
    - It will do this to the y coordinates too.
    - All of their will be done using if statements.
    - The code to set the new position will be written directly after the if statements.

Collectibles

- **setSprite**
    - The sprites for the collectibles will be set in this function. Sprites will be drawn after an asteroid has been shot at and undrawn after collection.
- **setValue**
    - An integer value will be assigned to each collectible. This value will be used to alter the player's space credits.