**Name:** Taurean Roberts

**Date:** December 6, 2017

**Assessment:** Maths for Games

1. **Requirements Documentation**
   1. **Description of Problem**

**Name:** Bootstrap

**Problem Statement:** Create a game using a custom linked library

**Problem Specifications:** Use linked library and made it

* 1. **Input Information**

The user is to use the up and down arrows on the keyboard to move the ship and avoid asteroids for a total of two minutes

* 1. **Output Information**

The game outputs the total time it takes to the “end” of the game.

* 1. **User Interface**

The game has a display that tells the user that they can exit at anytime when the game is running but pressing the escape key.

1. **System Architecture**
   1. **Entity Class:**

class Entity

{

public:

//Prototype: Entity()

//Description: Handles the Vector/Matrix classes when its child class is called

//Precondition: handles the Vector/Matrix classes

//Post Condition: returns places for used calls

//Protection Level: Public

Entity();

//Prototype: ~Entity()

//Description: Deconstructs the information given on in the default

//Precondition: Takes in the default construture

//Post Condition: deallocates memory

//Protection Level: Public

~Entity();

//Prototype: Vector2 mScale

//Description: Sets the size of a object or item

//Precondition: takes in a X and Y scale to size upon

//Post Condition: Returns its size given

//Protection Level: Public

Vector2 mScale;

//Prototype: Vector2 mPos

//Description: Gives cordinates to use for placement of debris and

player classes

//Precondition: takes in two values and gives a X & Y axis for placement

//Post Condition: returns the positon

//Protection Level: Public

Vector2 mPos;

};

* 1. **Player2D Class:**

#pragma once

#include "Entity.h"

#include "Debris.h"

class Player2D : public Entity

{

public:

//Prototype: Player2D()

//Description: Default constructor for player variables needed for game functions

//Precondition: Sets the position and size of the ship to be feed in to the game

//Post Condition: returns the set value when called in the function

//Protection Level: Public

Player2D();

//Prototype: ~Player2D()

//Description: Deconstructs the Player variables for Player2D functionality

//Precondition: Takes in the functionality of the default constructor

//Post Condition: Frees ram in the default

//Protection Level: Public

~Player2D();

//Prototype: void Update(float deltaTime)

//Description: Updates the player's position, speed, and boundries

//Precondition: gives the player's ship its position, speed, and boundries

//Post Condition: Feeds the information into the main function and allows movement properties

//Protection Level: Public

void Update(float deltaTime);

//Prototype: bool isAlive

//Description: the condition controls the losing condition when the player is dead it causes game over status

//Precondition: checks to see if the player is still alive

//Post Condition: if the player doesnt make the time limit and dies, game over

//Protection Level: Public

bool isAlive;

};

* 1. **Debris Class:**

#pragma once

#include "Entity.h"

#include "time.h"

class Debris : public Entity

{

public:

//Prototype: Debris().

//Description: Default constructor that holds all determined set values.

//Precondition: Sets values of the debris and gives it its size and position.

//Post Condition: returns the values to the shape when its name is called.

//Protection Level: Public.

Debris();

//Prototype: Debris(Vector2 pos, Vector2 scale);

//Description: Calls in the math .lib Vector2 class and gives additional output to the debris size.

//Precondition: takes in a argument for the position and the scale to be seeded in the spawning process.

//Post Condition: Gives the game a value to go to when the astroids hit the edge of the screen to reset to.

//Protection Level: Public

Debris(Vector2 pos, Vector2 scale);

//Prototype: ~Debris()

//Description: Deconstructs the default contructor to allocate memory

//Precondition: DeAllocates the variables called in the default constructors

//Post Condition: Frees up memory in default constructor

//Protection Level: Public

~Debris();

//Prototype: bool Move(Float deltaTime)

//Description: Gives the debris information on movement, direction, and speed when called

//Precondition: Takes in a float as a arguement

//Post Condition: returns the given movement, direction, and speed

//Protection Level: Public

bool Move(float deltaTime);

//Prototype: void resetRock(float deltaTime)

//Description: Gives the astroids its reset areas to spawn and despawn from

//Precondition: Hardsets the position for the X axis and randomizes the Y axis.

//Post Condition: Gives the Y axis new locations for the astroids to spawn from

//Protection Level: Public

void resetRock(float deltaTime);

};

1. **Source Code**
   1. **Entity class**

#include "Entity.h"

Entity::Entity()

{

}

Entity::~Entity()

{

}

* 1. **Player2D Class**

#include "Player2D.h"

#include <Input.h>

#include "Debris.h"

#include "Application2D.h"

Player2D::Player2D()

{

mPos.mX = 120;

mPos.mY = 400;

mScale.mX = 100;

mScale.mY = 100;

isAlive = true;

}

Player2D::~Player2D()

{

}

void Player2D::Update(float deltaTime)

{

aie::Input\* input = aie::Input::getInstance();

Vector2 playerPos(mPos.mX, mPos.mY);

// use arrow keys to move ship

if (input->isKeyDown(aie::INPUT\_KEY\_UP))

mPos.mY += 500.0f \* deltaTime;

if (input->isKeyDown(aie::INPUT\_KEY\_DOWN))

mPos.mY -= 500.0f \* deltaTime;

//Ship Boundries

if (mPos.mY > 690)

mPos.mY = 689;

if (mPos.mY < 30)

mPos.mY = 31;

}

* 1. **Debris class**

#include "Debris.h"

Debris::Debris()

{

mPos.mX = 600;

mPos.mY = 450;

mScale.mX = 100;

mScale.mY = 100;

}

Debris::Debris(Vector2 pos, Vector2 scale)

{

mPos = pos;

mScale = scale;

}

Debris::~Debris()

{

}

bool Debris::Move(float deltaTime)

{

float someSpeed;

for (int i = 0; i < 20; i++)

{

someSpeed = (rand() % ((35+1) - 10)+ 10);

mPos.mX -= someSpeed \* deltaTime;

}

return true;

}

void Debris::resetRock(float deltaTime)

{

mPos.mX = 1280;

mPos.mY = rand() % 720;

}

1. **Read Me**

When you are accessing this code there are things that you should know. This program needs to have a linked static library that uses vectors and matrixes, but mainly vectors in this project in order to be ran.