Name:

Date:

Assessment:

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

**Name:** Boot Strap Snake

**Problem Statement:** (Make a game using you Static Library)

**Problem Specifications:** (Make a basic game that Uses your Static Library with no errors)

* 1. **Input Information**

(The Application file and the Mouse and Keyboard)

* 1. **Output Information**

(There is no Audio output but there is visual output)

* 1. **User Interface**

(Press the Esc Key to exit the application.)

1. **System Architecture**

**Snake.h**

**Prototype:** (float SnakePos(Vector2 pos);)

**Description:** (sets the postion for the snake to a float.)

**Precondition:** (Needs to be correctly linked with Math Library )

**Post condition:** (the Snakes position is returned.)

**Protection Level:** (public)

**Prototype:** (void changeDirection(int Up,int Down,int Left,int Right))

**Description:** (A function that makes it possible for the object identified as the snake to switch directions.)

**Precondition:** (the snake is face a certain direction)

**Post condition:** (the snake’s facing direction is changed)

**Protection Level:** (public)

**Prototype:** (void move(int Up, int Down, int Left, int Right))

**Description:** (A functon that makes it possible for the object identified as the snake to move through the game board.)

**Precondition:** (the snake is not able to move)

**Post condition:** (the snake is able to move left, right, up, or down now)

**Protection Level:** (public)

**Grid.h**

**Prototype:** (int Grid(int Cols, int Rows))

**Description:** (a function that assigns that the grid is made of Columns and Rows.)

**Precondition:** (I. Ds that the grid is made of rows and columns )

**Post condition:** (The playing grid is made on grid thanks to the rows and columns acting as vectors)

**Protection Level:** (public)

**Prototype:** (int DrawGrid())

**Description:** (a function that draws an invisible grid that is placed on the inside of the game board and sets boarders.)

**Precondition:** (there is no invisible grid to play on)

**Post condition:** (the grid is inserted in the game so that movement can be grid based)

**Protection Level:** (public)

1. **Source Code**
2. **Application2D.h**
3. #pragma once
4. #include "Application.h"
5. #include "Renderer2D.h"
6. #include"Snake.h"
7. #include<Vector2.h>
8. #include<Vector3.h>
9. #include<Vector4.h>
10. #include<Matrix3.h>
11. #include<Matrix4.h>
12. #include<Matrixs2.h>
13. #include<iostream>
14. #include<math.h>
15. class Application2D : public aie::Application {
16. public:
18. Application2D();
19. virtual ~Application2D();
20. virtual bool startup();
21. virtual void shutdown();
22. virtual void update(float deltaTime);
23. virtual void draw();
24. protected:
25. aie::Renderer2D\* m\_2dRenderer;
26. aie::Texture\* m\_texture;
27. aie::Texture\* m\_shipTexture;
28. aie::Font\* m\_font;
29. Snake \*mSnake;
30. float m\_cameraX, m\_cameraY;
31. float m\_timer;
32. };
33. **Application2D.cpp**

#include "Application2D.h"

#include "Texture.h"

#include "Font.h"

#include "Input.h"

#include<Vector2.h>

#include<Vector3.h>

#include<Vector4.h>

#include<Matrix3.h>

#include<Matrix4.h>

#include<Matrixs2.h>

#include<iostream>

#include<math.h>

Application2D::Application2D() {

}

Application2D::~Application2D() {

}

bool Application2D::startup() {

m\_2dRenderer = new aie::Renderer2D();

m\_texture = new aie::Texture("./textures/numbered\_grid.tga");

/\*m\_shipTexture = new aie::Texture("./textures/ship.png");\*/

m\_font = new aie::Font("./font/consolas.ttf", 32);

mSnake = new Snake(Vector2(5,5));

mSnake->mPos.mY = 5;

m\_cameraX = 0;

m\_cameraY = 0;

m\_timer = 0;

return true;

}

void Application2D::shutdown() {

delete m\_font;

delete m\_texture;

//delete m\_shipTexture;

delete m\_2dRenderer;

}

void Application2D::update(float deltaTime) {

m\_timer += deltaTime;

// input example

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

if (input->isKeyDown(aie::INPUT\_KEY\_UP))//to move up

mSnake->changeDirection(1, 0, 0, 0);

if (input->isKeyDown(aie::INPUT\_KEY\_DOWN))// to move down

mSnake->changeDirection(0, 1, 0, 0);

if (input->isKeyDown(aie::INPUT\_KEY\_LEFT))//to move left

mSnake->changeDirection(0, 0, 1, 0);

if (input->isKeyDown(aie::INPUT\_KEY\_RIGHT))//to move right

mSnake->changeDirection(0, 0, 0, 1);

// exit the application

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

quit();

}

void Application2D::draw() {

// wipe the screen to the background colour

clearScreen();

// set the camera position before we begin rendering

m\_2dRenderer->setCameraPos(m\_cameraX, m\_cameraY);

// begin drawing sprites

m\_2dRenderer->begin();

// demonstrate animation

/\*m\_2dRenderer->setUVRect(int(m\_timer) % 8 / 8.0f, 0, 1.f / 8, 1.f / 8);

m\_2dRenderer->drawSprite(m\_texture, 200, 200, 100, 100);\*/

//// demonstrate spinning sprite

//m\_2dRenderer->setUVRect(0,0,1,1);

//m\_2dRenderer->drawSprite(m\_shipTexture, 600, 400, 0, 0, m\_timer, 1);

// draw a thin line

m\_2dRenderer->setRenderColour(1, 0, 1, 1);

m\_2dRenderer->drawBox(mSnake->mPos.mX, mSnake->mPos.mY, 1, 1);

m\_2dRenderer->setRenderColour(1, 0, 0, 1);

//Borders

m\_2dRenderer->drawLine(0, 0, 1280, 0, 30, 30);

m\_2dRenderer->drawLine(1280, 720,1280 , 0, 30, 1);

m\_2dRenderer->drawLine(1280, 720, 0, 720, 30, 1);

m\_2dRenderer->drawLine(0, 720, 0, 0, 30, 1);

//// draw a moving purple circle

//m\_2dRenderer->setRenderColour(1, 0, 1, 1);

//m\_2dRenderer->drawCircle(sin(m\_timer) \* 100 + 600, 150, 50);

//// draw a rotating red box

//m\_2dRenderer->setRenderColour(1, 0, 0, 1);

//m\_2dRenderer->drawBox(600, 500, 60, 20, m\_timer);

//// draw a slightly rotated sprite with no texture, coloured yellow

m\_2dRenderer->setRenderColour(1, 1, 0, 1);

m\_2dRenderer->drawSprite(nullptr, 25, 25, 25, 25, 3.14159f \* 0.25f, 1);

// output some text, uses the last used colour

char fps[32];

sprintf\_s(fps, 32, "FPS: %i", getFPS());

m\_2dRenderer->drawText(m\_font, fps, 0, 720 - 32);

m\_2dRenderer->drawText(m\_font, "Press ESC to quit!", 0, 720 - 64);

// done drawing sprites

m\_2dRenderer->end();

}

**Snake.h**

#pragma once

#include<Vector2.h>

#include<Vector3.h>

#include<Vector4.h>

#include<Matrix3.h>

#include<Matrix4.h>

#include<Matrixs2.h>

#include<iostream>

#include<math.h>

class Snake

{

public:

int mSnake;//Creates a global varaible called mSnake.

Snake();

Snake(Vector2 pos);//Creates an instance for the snake to have a pos using the pos constructor from the Vector2 class.

Vector2 mPos;//Creates an instance for the variable mPos.

float SnakePos(Vector2 pos);//sets the postion for the snake to a float.

int headxpos;//Sets the mX positon of the snakes head.

int headypos;//Sets the mY position of the snakes head.

int direction;//used in the function change direction to identfy a direction.

// Moves snake head to new location

void changeDirection(int Up,int Down,int Left,int Right);//A function that makes it possible for the object identified as the snake to switch directions.

void move(int Up, int Down, int Left, int Right);//A functon that makes it possible for the object identified as the snake to move through the game board.

};

**Snake.cpp**

#include "Snake.h"

#include<Vector2.h>

#include<Vector3.h>

#include<Vector4.h>

#include<Matrix3.h>

#include<Matrix4.h>

#include<Matrixs2.h>

#include<iostream>

#include<math.h>

Snake::Snake()

{

int mSnake;

}

Snake::Snake(Vector2 pos)

{

mPos = pos;

}

float Snake::SnakePos(Vector2 pos)

{

mPos.mX = pos.mX;

mPos.mY = pos.mY;

return mPos.mX&&mPos.mY;

}

void Snake::changeDirection(int Up, int Down, int Left, int Right)

{

if (Up == 1)

mPos = mPos + Vector2(0, 50);

if (Down == 1)

mPos = mPos + Vector2(0, -50);

if (Left == 1)

mPos = mPos + Vector2(-50, 0);

if (Right == 1)

mPos = mPos + Vector2(50, 0);

}

void Snake::move(int Up, int Down, int Left, int Right)

{

}

**Grid.h**

#pragma once

class PlayingArea

{

public:

int mRows;//Creates a varaible for the number of rows in the game board.

int mCols;//Creates a varaible for the number of colmons in the game board.

const int mapwidth = 1280;//sets a constant measurement for the width of the game board.

const int mapheight = 1280;//sets a constant measurement for the heigth of the game board.

int Grid(int Cols, int Rows);//a function that assigns that the grid is made of Colmons and Rows.

int DrawGrid();//a function that draws an ivisiable grid that is placed on the inside of the game board and sets boarders.

};

**Grid.cpp**

#include"Grid.h"

#include<Vector2.h>

#include<Vector3.h>

#include<Vector4.h>

#include<Matrix3.h>

#include<Matrix4.h>

#include<Matrixs2.h>

#include<iostream>

#include<math.h>

int PlayingArea::Grid(int Cols, int Rows)

{

PlayingArea mGrid;

mGrid.mRows = Rows;

mGrid.mCols = Cols;

return Rows&&Cols;

}

int PlayingArea::DrawGrid()

{

PlayingArea mGrid;

mGrid.mRows = mRows;

mGrid.mCols = mCols;

return mGrid.mRows&&mCols;

}

**Read Me**

(Be very clear as to how the assessor should go about getting your application, running it, and using it. You should assume the assessor knows nothing about your application.)

* **Step1:**

Go to the GitHub website search for the username Ralenski Doucet. Once the user is found go to the user’s repositories and open the repository Math for Games. Then Download a zip copy of the repository. Then go your computers downloads folder and select/open the file you have just downloaded in Microsoft Visual studio 2015.

* **Step2:**

Once the file is open go to the solution window and right click on the mouse and the select the option that says “Properties”. Once the window for Properties window is open select the tab that says “General”.

Then select the drop down tab that says “static library”. Then switch from static library to application(.exe).

Then select apply and then ok to save those changes. Then go back to your downloaded files and open the file that we opened earlier in Microsoft Visual Studios 2015.

* **Step3:**

To end the Program, reopen the application that we minimized earlier and press the X to permanently close the application and then while in visual studios Press and hold the shift key at the same time as the F5 key.

Then to close and exit out of Microsoft Visual Studios Press the X button with the mouse found in the top right corner of the screen.