DigiPen Institute of Technology
CSD1130
Game Implementation techniques
Assignment 3 – Part 1
Platform Functionalities

Due date:
Wednesday, Feb 22nd, 2023, at 2:00pm

Topics
The assignment will cover the following topics:

1. Importing map data from a file.
2. Binary map collision check.
3. Object snapping.

Goal
The goal of this assignment is to implement the main functionalities needed for a platform game. These functionalities will be used in the 2nd part of this assignment, which is implementing a platform game.

DigiPen Institute of Technology
CSD1130
Game Implementation techniques
Assignment 3 – Part 1
Platform Functionalities

Submission Guidelines – Visual Studio [Using Microsoft C++ Compiler]

- Submit at Moodle entry:
  "**csd1130_Assignment_3_Part1_Submission_VisualStudio**"
- To submit your programming assignment, organize a folder consisting of BinaryMap.cpp file **ONLY**. Name this folder, with only small letters, using the following convention:

  <class>_<section>_<student login name>_<assignment#>_<part#>

  For example, if your login is ***foo.boo*** and assignment 1 part 1 is being submitted, your folder would be named **csd1130_a_foo.boo_1_1**
- Your folder must not contain any extra file.
- Zip this folder and name the resulting file using the following convention:

  <class>_<section>_<student login name>_<assignment#>_<part#>.zip

  For example, if your login is ***foo.boo*** and you are submitting assignment 1 part 1, your zipped file would be named as: **csd1130_a_foo.boo_1_1.zip**
- Next, upload your zip file after logging into the course web page using the link https://distance3.sg.digipen.edu.

- Finally, perform a sanity check to determine if your programming submission follows the guidelines by downloading the previously uploaded zip file, unzipping it, then compiling, linking, and executing your submission.

Using the right Compiler and MSVS setup

- The project must be tested in **RELEASE and DEBUG** modes with **warning level 4**, under **x64** platform. Under project settings the **SDK Version**: must be set to *10.0 (latest installed version)* and the **Platform Toolset** set to *Visual Studio 2019 (v142)*. It must generate 0 warnings and 0 errors. This can be verified on a PC located at **Pascal lab**.

- Please validate the previous statement before your submission!

DigiPen Institute of Technology
CSD1130
Game Implementation techniques
Assignment 3 – Part 1
Platform Functionalities

Submission Guidelines – VPL [Using g++ Compiler under A3_P1 – VPL submission]

- Submit under this VPL entry, after you finish and submit your work under the Visual Studio entry!
- Submit at Moodle entry: "**csd1130_Assignment_3_Part1_Submission_VPL**"
- To submit your programming assignment:
    - Upload your 'BinaryMap.cpp' code file, done in the visual studio project
    - Run & Evaluate
        - Correct evaluation will give the following result:
            - **Summary of tests**
            - [20 test run/ 20 test passed]

DigiPen Institute of Technology
CSD1130
Game Implementation techniques
Assignment 3 – Part 1
Platform Functionalities

Description [Using Microsoft C++ Compiler]

- ✓ This project must be implemented as a Win32 **Console** Application (*start with an empty project – Do not include precompiled headers support*)

- ✓ Language: C/C++

- ✓ A main.cpp file is provided. (**Do not modify this file!**)
  - This file is a driver which will test your implemented functions.

- ✓ A BinaryMap.h file is provided. (**Do not modify this file!**)
  - This file contains the binary map functions declarations.

- ✓ A BinaryMap.cpp file is provided.
  - It includes the variables needed by the Binary Collision Map.
    - ➢
```
/*The number of horizontal elements*/
static int BINARY_MAP_WIDTH;
```
    - ➢
```
/*The number of vertical elements*/
static int BINARY_MAP_HEIGHT;
```
    - ➢
```
/*This will contain all the data of the map, which will be retrieved from a file when the "ImportMapDataFromFile" function is called*/
static int **MapData;
```
    - ➢
```
/*This will contain the collision data of the binary map. It will be filled in the "ImportMapDataFromFile" after filling "MapData". Basically, if an array element in MapData is 1, it represents a collision cell, any other value is a non-collision*/
static int **BinaryCollisionArray;
```

- ✓ An exported file will be provided.
  - Use the provided exported file to test your project.

- ✓ You must implement a function to import map data from a file.
  - Prototype: int **ImportMapDataFromFile**(char *FileName)
    - ➢ FileName: Name of the file to retrieve data from.
    - ➢ Returned value: 1 if the function succeeds, 0 if it does not (Example: the file does not exist).

- Description:
  - This function opens the file name "FileName" and retrieves all the map data.
  - It allocates memory for the 2 arrays: MapData & BinaryCollisionArray.
  - The first line in this file is the width of the map.
  - The second line in this file is the height of the map.
  - The remaining part of the file is a series of numbers.
  - Each number represents the ID (or value) of a different element in the double dimensional array.

  - Example:
    Width 5
    Height 5
    1 1 1 1 1
    1 1 1 3 1
    1 4 2 0 1
    1 0 0 0 1
    1 1 1 1 1

    After importing the above data, "MapData" and " BinaryCollisionArray" should be:

    1 1 1 1 1
    1 1 1 3 1
    1 4 2 0 1
    1 0 0 0 1
    1 1 1 1 1

    and
    1 1 1 1 1
    1 1 1 0 1
    1 0 0 0 1
    1 0 0 0 1
    1 1 1 1 1

    respectively.

- Finally, the function returns 1 if the file named "FileName" exists, otherwise it returns 0.

- The file named “FileName” is a .txt file generated by the provided map editor (Level_Tool.exe).

✓ You must implement a function that frees the memory that was allocated for the binary map.
- Prototype: void **FreeMapData**(void);

- Description: This function frees the memory that was allocated for MapData and BinaryCollisionArray in the “ImportMapDataFromFile” function.

✓ You have to implement a function that retrieves a cell's ID (value) from the two-dimensional array BinaryCollisionArray.
- Prototype: int **GetCellValue**(int X, int Y);
  - X: The x component of the cell you want to check.
  - Y: The y component of the cell you want to check.
  - Returned value: The value of the cell whose coordinates are X and Y from the BinaryCollisionArray array.

- Description:
  - Before retrieving the value, it should check that the supplied X and Y values are not out of bounds (in that case return 0).

✓ You must implement a function that checks for collision between an object instance and the binary collision map.
- Prototype: int **CheckInstanceBinaryMapCollision**(float PosX, float PosY, float scaleX, float scaleY);
  - PosX: The x position of the object instance.
  - PosY: The y position of the object instance.
  - scaleX: Width of the object instance.
  - scaleY: Height of the object instance.
  - Returned value: A flag where each bit represents 1 collision side (Check description).

- Description:
  - This function creates 2 hot spots on each side of the object instance.
  - It checks if each of these hot spots is in a collision area.
  - At the beginning of the function, a "Flag" integer should be initialized to 0.
  - Each time a hot spot is in a collision area, its corresponding bit in "Flag" is set to 1.
    - The collision sides bits are defined as follows:

```
const int   COLLISION_LEFT      = 0x00000001;   //0001
const int   COLLISION_RIGHT     = 0x00000002;   //0010
const int   COLLISION_TOP       = 0x00000004;   //0100
const int   COLLISION_BOTTOM    = 0x00000008;   //1000
```

  - Note: This function assumes the object instance's size is 1 by 1 (the size of 1 cell).
  - Finally, the function returns the integer "Flag".

  - Example: Creating the hotspots.
    - Handle each side separately.
    - 2 hot spots are needed for each collision side.
    - These 2 hot spots should be positioned on 1/4 above the center and 1/4 below the center

    - Example: Finding the hots spots on the right side of the object instance

```
float x1, y1, x2, y2;

hotspot 1
x1 = PosX + scaleX/2     To reach the right side
y1 = PosY + scaleY/4     To go up 1/4 of the height

hotspot 2
x2 = PosX + scaleX/2     To reach the right side
y2 = PosY - scaleY/4     To go down 1/4 of the height
```

- ✓ You have to implement a function that snaps a value to the center of the cell it's contained in.
  - Prototype: void **SnapToCell**(float *Coordinate);
    - Coordinate: Address of the float value that the function will snap to the center of the cell it is in.

  - Description:
    - To snap "Coordinate" to the center of the cell, first find its integral part.
    - Add 0.5 to the integral part in order to position it in the middle of the cell.
      - Remember that the cell's dimensions are always (1,1) in the normalized coordinates system.
    - Save the result back in "Coordinate".

- ✓ A function name "**PrintRetrievedInformation**" is provided.
  - This function prints out the content of the 2D array "MapData", as it was loaded from "Exported.txt" file, to the console.
  - Use this function to make sure the information you retrieved from the .txt file is correct.

Note:

- You are supposed to submit only "**BinaryMap.cpp**" file but do not forget to compile in Warning level 4.
- Make sure **NOT** to change your BinaryMap.h file because you are submitting **only** BinaryMap.cpp.
- You are not allowed to add any extra function or any global variable other than the given ones.
- A tile editor (Level_Tool.exe) is provided to test a different map size.

DigiPen Institute of Technology
CSD1130
Game Implementation techniques
Assignment 3 – Part 1
Platform Functionalities

Evaluation

Here are the most common reasons assignments are marked down:

- Project does not build.
- Project does not build without warnings.
- One or more items in the "Requirements" section was not satisfied.
- A fundamental concept was not understood.
- Code is sloppy and hard to read (e.g. indentation is not consistent, no comments, etc.…)
- Your solution is difficult (or impossible) for someone reading the code to understand due to lack of comments, poor variable/method names, poor solution structure, etc.…
- Project assignment was turned in late.

DigiPen Institute of Technology
CSD1130
Game Implementation techniques
Assignment 3 – Part 1
Platform Functionalities

Grading Algorithm

This project will be graded according to the following rules. Any rule that is missing, points will be deducted from the project's grade:

- Compile errors or does not compile for some reasons.
- Error/Crash (on loading and/or running a different sized map – different than the one provided)
    - To validate this rubric, as a side test, create, load and test 2 separate maps, with non-equal sides (5x7) and (7x5), and make sure the arrays in memory are built correctly and they can free correctly! This will validate that you did not mix-up your width and height variables of the map in your loops.
- Compile Warnings.
- Did not check for out of bound values, in "GetCellValue()".
- "PrintRetrievedInformation" is not correct.
- Output not matching the given one(s).
- If having Memory leaks!

Notes for Moodle submissions:

- "Visual Studio" submission is **NOT** counted for A3_P1 grading!
- "VPL" submission is counted as the full score for A3_P1 grading.