Fall 2013
CS 529 | Fundamentals of Game Development
Project 2 | Part 1 – Platformer Functionalities

Files (submit folder) due

- Sunday, September 29th, 2013
- 11.55pm

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 platformer game. These functionalities will be used in the 2nd part of this project, which is implementing a platformer game.

Assignment Submission

- Compress (.zip) the solution folder (Delete the debug/release folders and the .ncb file first), and submit it on distance.digipen.edu.

Check the course syllabus regarding the naming and submission convention.

Description

- This project has to be implemented as a Win32 Console Application

- Language: C

- A main.c file is provided.
  - This file is a driver which will test your implemented functions.

- A BinaryMap.h file is provided
  - This file contains the binary map functions declarations.

- A BinaryMap.c 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 cell*/
      static int **BinaryCollisionArray;
      ```

- A map data file will be provided (Exported from the editor).
  - Use the provided exported file to test your project.

- You have to 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 doesn't (Example: the file doesn't exist).
  - Descripion:
    - This function opens the file name "FileName" and retrieves all the map data.
    - It allocates memory for the 2 arrays: MapData & BinaryCollisionArray.
      - **Dynamic memory allocation!**
    - 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 0 0 0 1
  1 4 2 0 1
  1 1 1 3 1
  1 1 1 1 1

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

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

  and

  1 1 1 1 1
  1 0 0 0 1
  1 0 0 0 1
  1 1 1 0 1
  1 1 1 1 1

  respectively.

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

  - 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 have to 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 whose 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 (which means the cell if falls in has a value of 1).
    - 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:

```
#define   COLLISION_LEFT     0x00000001   //0001
#define   COLLISION_RIGHT    0x00000002   //0010
#define   COLLISION_TOP      0x00000004   //0100
#define   COLLISION_BOTTOM   0x00000008   //1000
```

    - Note: This function assumes the object instance's size is 1 by 1 (the size of 1 tile).
    - 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 hot spots on the left side of the object instance.

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

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

hotspot 2
x2 = PosX - scaleX/2      To reach the left 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's 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"
  - Use this function to make sure the information you retrieved from the .txt file is correct.

- Finally, each ".c" and ".h" file in your homework should include the following header:

```
/ *-----------------------------------------------------------------------------
Project Title           :       CS 230: Project 3 Part 1
File Name               :       (Enter file name here)
Author                  :       (Enter your name here)
Creation Date           :       (Enter the creation date of the file)
Purpose                 :       (Enter the main purpose of the file here)
History
 -(Enter date here)     :       (Enter modifications done on current date here)
 -(Enter date here)     :       (Enter modifications done on current date here)
© Copyright 1996-2011, DigiPen Institute of Technology (USA). All rights reserved.
-----------------------------------------------------------------------------*/
```