Project Proposal for CG 100433 course

Project Title

BomberMan（single version）

Team member

student number	name

Abstract

In this project, we use CG techniques to implement 'BomberMan', a classic game which is suitable for project's requirement. It is hoped to complish basic game logic with consice UI design and fine CG effect. Using related CG techniques like 3D model loading, multiple light rendering, shadow mapping based on SDL and openGL, we achieve our goals finally.

Motivation

The idea comes from our childhood game —— BomberMan. Every team member is familiar with this classic game and we are excited to complish it in our own way using knowledge from this class. Besides, easy game logic and moderate complexity are in our ability. Numerous open-source models related to this game also help us a lot.

The Goal of the project

1. Achieve players' movement and basic posture.
2. Achieve bomb's placing and explosion
3. Player can compete with enemies, and the game can automatically give a result according to game situation.
4. Achieve the basic light effect, like shadow of player and enemies, objects
5. Achieve shape change of bombers and objects under explosion and light effect of explosion
6. Removable perspective implementation

The Scope of the project

1. loading 3D model
2. scene modeling and map generation

3. multi-source rendering
4. light movement
5. physic engine will not conclude
6. shadows of models

Related CG technique

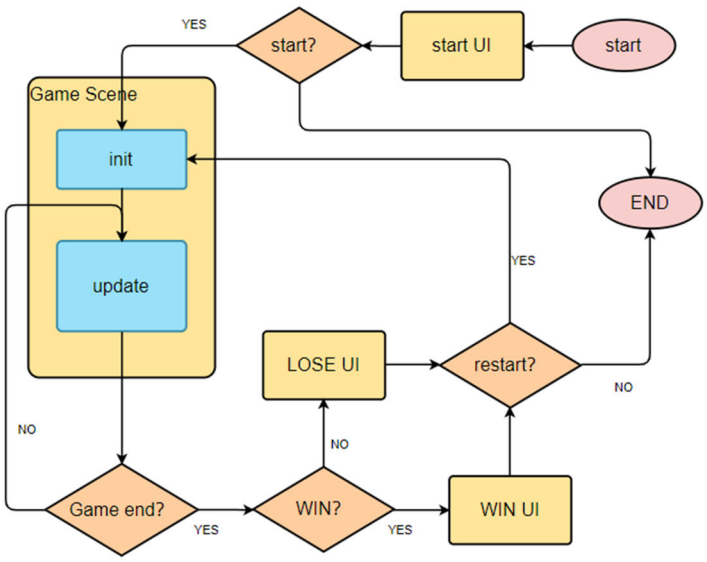
1. modeling
2. viewing
3. transformation
4. rendering
5. shadow mapping

Project contends

1. start/win/lose UI interface
2. map generation, using keyboard to control player.
3. movement of player and enemies. Both only can place a bomber in one square.
4. After specific period, bomber will explode, player or enemies within explosive area will dead, the objects will be destroyed.
5. Pathfinding algorithm is included for enemies' movement. enemies also can put bombers automatically.
6. Decision of the game: Players win only when all the enemies died because of bomber explosion. Conversely, if player dies of explosion, game over.
7. The effect of shadow of objects, multisource light.
8. The effect of bomber explosion.

Implementation

Game logic:



whole process

1. Operation logic:

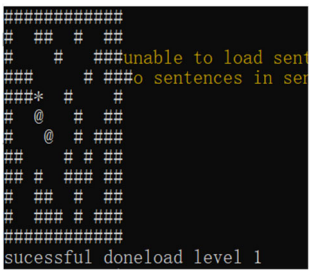
   There are three enemies each round, and you should control “up”,” down”,” left” and “right” keys to let your character move to avoid the bombs these enemies put. The player also can press the “space” key to put a bomb. Every bomb will explode within two seconds after it is placed. When a bomb explodes, any characters around the bomb will be killed, if you can kill all enemies, you win, but if you are killed by any of the bombs, you lose.

2. Implementation logic:

   A total of three classes are used in the completion of the game logic, the player class, the enemy class and the map class. They have different divisions of labor but are interconnected to form a whole game implementation. The game class is responsible for the control of the entire game. It contains the instantiation of the enemy class and the map class, records various real-time information of the player, and provides an interface for interaction, and provides the necessary parameters for real-time rendering of the model and screen. The design also considers the continuity processing required when rendering the enemy's movement, such as providing a bool value of whether an enemy is walking, the direction of walking, etc.

3. Interface interaction logic:

   As mentioned above, the game logic part provides the necessary parameters for real-time rendering models and screens. Among them, the continuous walking of the character is an important problem that needs to be solved. The underlying game logic can realize the automatic pathfinding of the enemy, and update it in real time every second, as showed follow,



   but for the display of the interface, we need to let the enemy slowly moves towards the next position. Therefore, the underlying game logic must meet the function of predicting the position, which means that when a new position is reached, the next position to be reached must also be determined. Then information about whether to walk and the direction of move are given to the drawing part.

Lighting and shadow:

As for lighting, using multiple lights (containing 1 directional light, 10 point lights, 1 spotlight) in a scene the approach is as follows: we have a single color vector that represents the fragment's output color. For each light, the light's contribution to the fragment is added to this output color vector. So each light in the scene will calculate its individual impact and contribute that to the final output color.

```
// define an output color value
  vec3 output = vec3(0.0);
```

```
// add the directional light's contribution to the
output
output += someFunctionToCalculateDirectionalLight();
// do the same for all point lights
for(int i = 0; i < point_lights_number; i++)
    output += someFunctionToCalculatePointLight();
// and add others lights as well (like spotlights)
   output += someFunctionToCalculateSpotLight();
```

Then based on this, add shadow mapping.

The first pass requires to generate a depth map. To generate the depth map simply,we only use one directional light to generate shadow, and moving this light's position to achieve the effect of light movement. Because based on directional light, we use an orthographic projection matrix for the light source where there is no perspective deform.

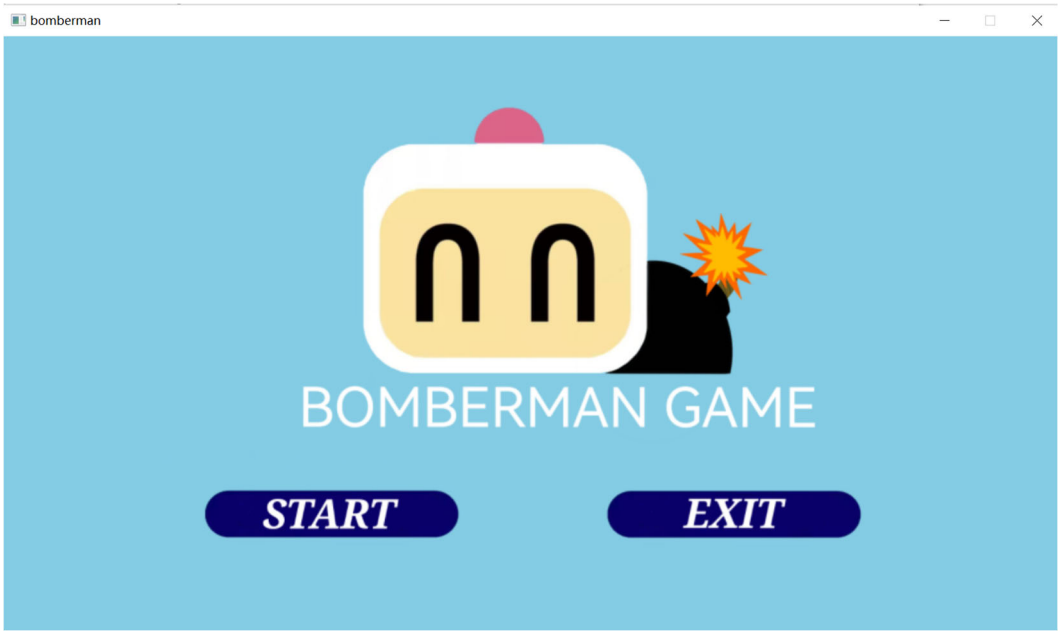
Next, rendering to depth map with onto a quad.

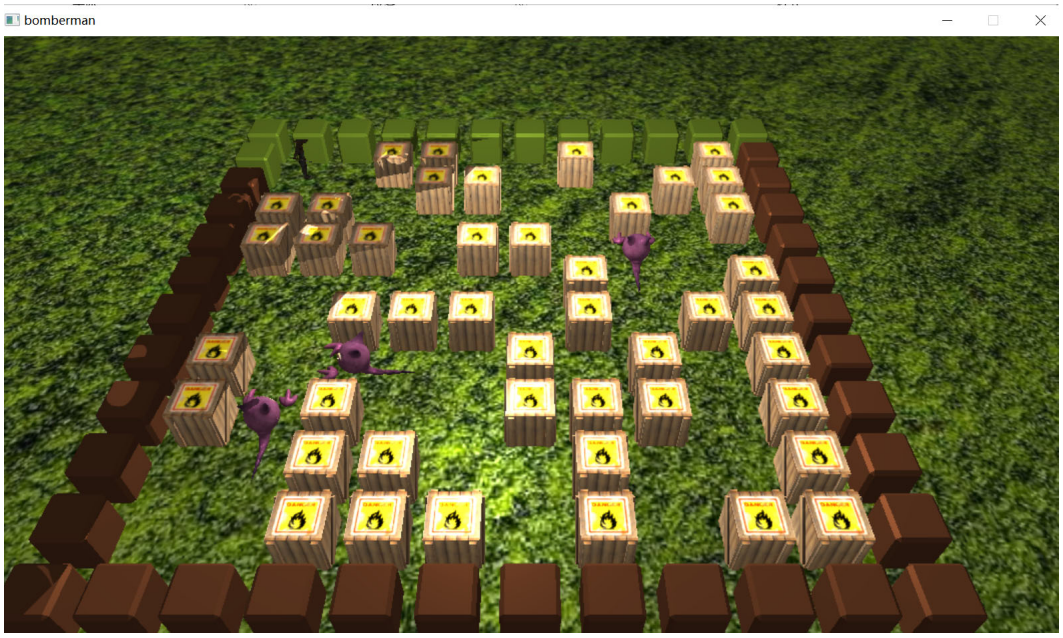
Finally, with a properly generated depth map, rendering the actual shadows. We do the light-space transformation in the vertex shader, check if a fragment is in shadow is (quite obviously) executed in the fragment shader.

PCF is also added to improve shadow maps.

For combing multiple source and shadow mapping, we choose weighted mean based on visual effect.

Results





Roles in group

work	name
Game logic (Find Path algorithm of enemies, decision of bomber placement, map generation, judgement of game)	
Model import & texture	
dynamic model animation	
Skybox implementation	

UI design	
The effect of explosion Project connection	
Game logic (changing interface logic, Gui design) The effect of shadow multiple light rendering & light movement	

Reference

- LearnOpenGL CN (learnopengl-cn.github.io)
- https://github.com/qwikdraw/bomberman
- tnicolas42/bomberman-assets at 85f00248d78a58e6318ebca8601b4e9f3295ff28 (github.com)
- Q 版泡泡堂小游戏,在线玩,4399 小游戏