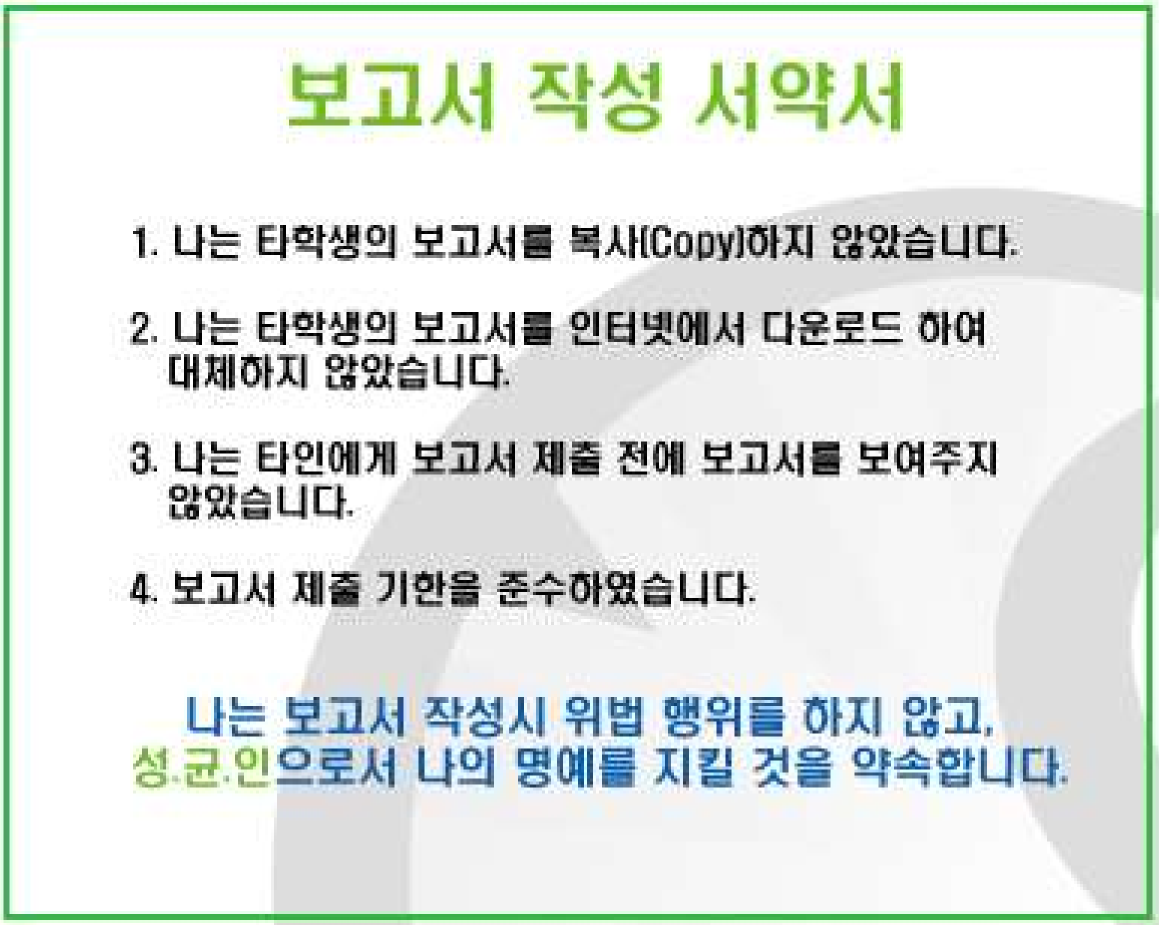
REPORT



보고서 작성 서약서

1. 나는 타학생의 보고서를 복사(Copy)하지 않았습니다.
2. 나는 타학생의 보고서를 인터넷에서 다운로드 하여 대체하지 않았습니다.
3. 나는 타인에게 보고서 제출 전에 보고서를 보여주지 않았습니다.
4. 보고서 제출 기한을 준수하였습니다.

나는 보고서 작성시 위법 행위를 하지 않고,
성.균.인으로서 나의 명예를 지킬 것을 약속합니다.

과 목 명	: Computer Graphics
담당교수	: 이 성 길
학 과	: 기계공학부
학 년	: 3
학 번	: 2017314380
이 름	: 윤 성 경
과 제 명	: T1 Team Project
제 출 일	: 2019. 12. 1

Algorithms and data structures

First, I made each object of game as independent struct in the header file to manage their properties and methods separately for better management. Their names are hero.h, bullet.h, circle.h, triangle.h, square.h, explosion.h. The header file contains attributes and methods like position, radius, initialization routine, update routine and revival routine. The global variables consists of the game objects, game modes, game speed and other states for managing the game flow control.

Second, I made the way how the objects interact to each other. The logic part on each frame render is composed of 1. Handling hero phase, 2. Handling bullets phase (containing crash test), 3. Handling enemy (circle, triangle, square) phase, 4. Handling particles phase, 5. Hit test phase. 6. Information phase

1. Handling hero phase smoothly moves the hero as user input controls, checks the skill status and blinks the hero in immortal situation.
2. Handling bullets phase assigns the inactive bullet for the request input, checks and handles the crash between the bullet and the active enemy.
3. Handling enemy phase manages the active enemy's action pattern and phase transition, checking the remaining time for inactive enemy to revive.
4. Handling particle phase manages active particles' remaining time.
5. Hit test phase checks if hero is hit by enemy using the equations involving the dot product (to check whether hero circle has crash with square). Also it checks if the game is over or not.
6. Information phase shows the help messages if the mode is help mode, calculates and shows the remaining skill cool time if the skill is not ready, showing the total score the player has earned.

Third, I will explain the data structure used for efficient rendering with many dynamic objects from diverse shapes. This has two parts, 1. Vertex, index and frame buffer structure. 2. Object pool structure.

1. For using the many kind of objects with different vertex and index buffer, I integrated them in the same shared vertex buffer and index buffer with different offset for each shape. This method is better than using many program method in performance, because it removes the program switching overhead between rendering the different shaped objects. We can also use different meshes in the same program by this method (it's valuable if the rendering is intensive). I also used separate program for rendering texts, but I think embedding the similar objects in a program is sometimes useful.
2. Object pool structure is the pre-allocated object set for effective management of objects switching activation frequently without memory assignment overhead. Because the objects are already and always allocated in a static space, object can freely activated and removed from the screen not changing the memory allocation. By removing the memory management overhead, this structure enabled more efficient rendering for many objects.

Advanced features not covered in the class + Discussion

I used multiple textures in rendering by loading image as the texture with stb_image.h. The images for texture are carefully selected and tuned to match the character properly. The texture are selected in texture's fragment file by the Boolean value, isTex0. I also mixed texture and original solid color of object itself with proper ratio in fragment for give them better visual illustration.

Although I used only few vertices and indices for simpler object processing and scalability, I used blinn-phong shading model which is meaningful with 3d-objects. Using appropriately tilted normal vector on each vertex, I could successfully represent better light effect with 2d like small vertices set.

I used text loading module named stb_truetype.h for print information in screen. It loads and creates the vertices and textures matching for each character font.

I used sound engine module named irrKlang and played background music and effect sounds. Finding the proper music and making reference was some work.

I carefully designed the game level and object action pattern. This required much time for play and feedback for game variable. It required a lot effort but it was fun.