Hackathon

I2P(II)_2019_SR

Q&A

- Use Slido for Q&A
- Join by the link: https://app2.sli.do/event/kfpgkfy8
  or by event code: #I2P2

Tower Defense

Mini Project 2 Package

Before we start,

Announcements

- You should have finished installing Allegro5 and set up your IDE on your own computer last semester in I2P course.
- If you did not take the course, see the Tutorial and videos.
- Our template requires Allegro5 and C++11 and you should compile and run the template successfully beforehand.
- If you use Visual Studio, you can download the project directly: Visual Studio Project Template

Outline

- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Outline

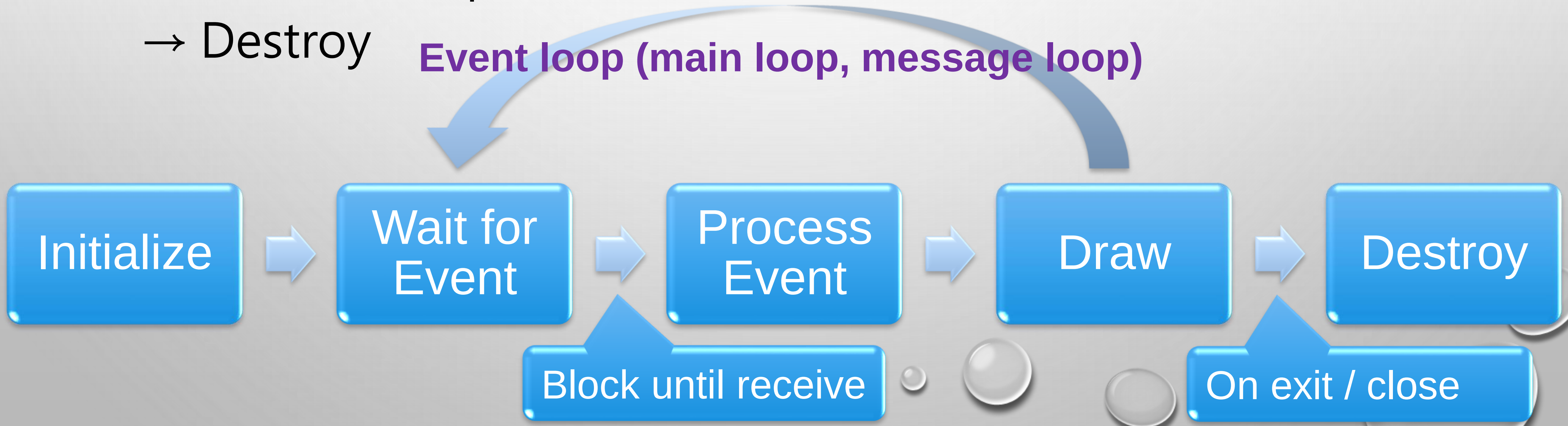
- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Quick Review

Allegro5 in C

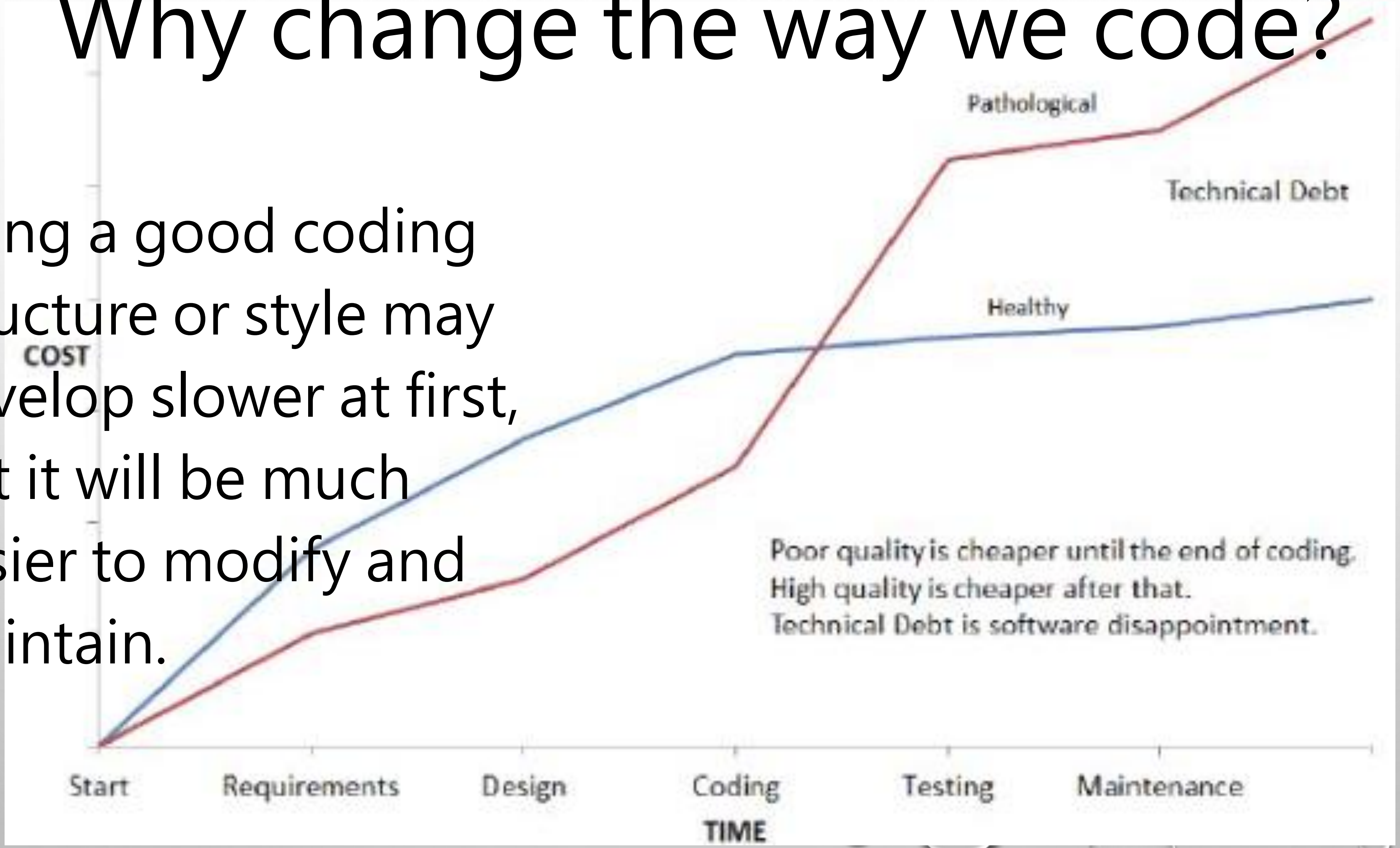
Program Flow in Allegro5

- Your codes are still sequential.
- Initialize → loop (Wait for event → Process event → Draw) → Destroy



Why change the way we code?

- Using a good coding structure or style may develop slower at first, but it will be much easier to modify and maintain.



Source: http://www.critical-logic.com/services/qa-project-management/

Quick Demo

TowerDefense game demo

What do we actually care?

and what we don’ t care?

Outline

- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Resources

- Specify only what type of resources and where can we load them.

Images (Bitmap)

Audios (Samples)

Fonts

Resources Management

- Manually loading / destroying resources is unnecessary and causes memory leak if we are not careful enough.

```
ALLEGRO_BITMAP* img = al_load_bitmap("img.png");
if (!img)
    game_abort("failed to load image: img.png");
//...
al_destroy_bitmap(img);
```

Resources Management

- We can ignore resource management when using the wrapped `Resources` class: more convenient and less error prone.

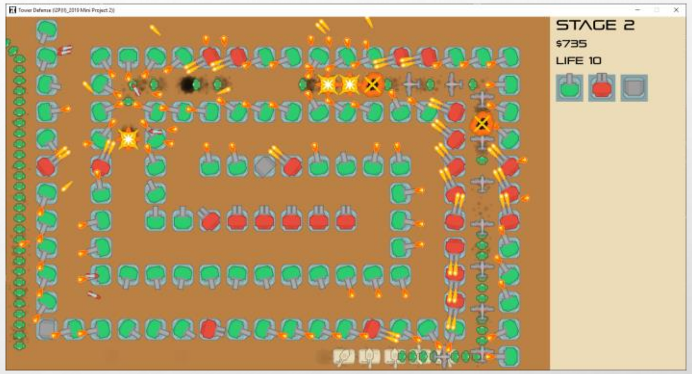
```
Resources::GetInstance().GetBitmap("img.png");
//...
// Automatically free resources.
```

Outline

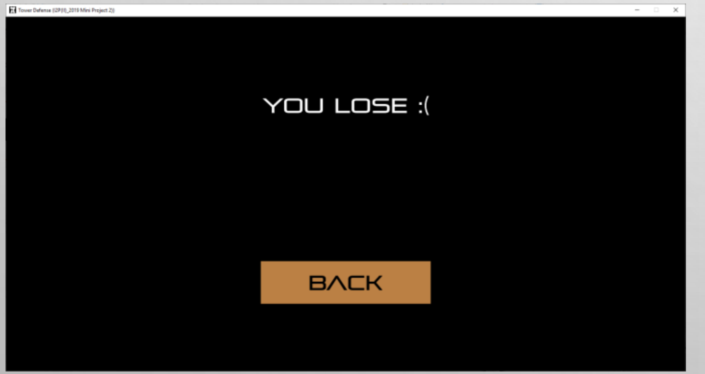
- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Scenes

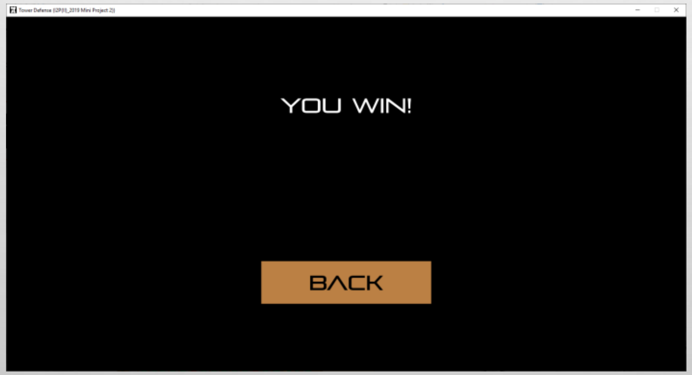
- All scenes should be independent.
- Change between scenes with only a function call.



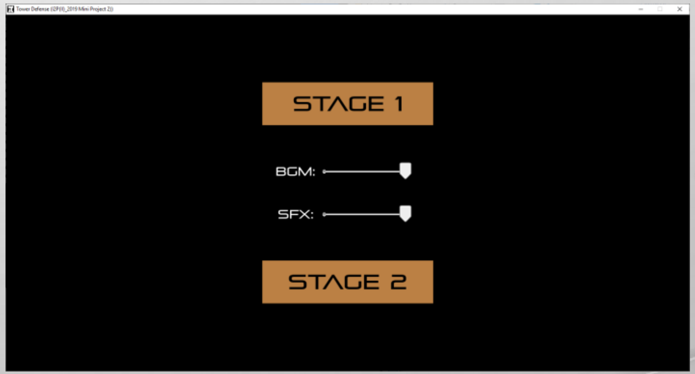
Play Scene



Lose Scene



Win Scene



Stage Select Scene

Multiple Scenes

- Manually checking which scene to update / draw is redundant and we cannot have same variable names in different scenes.

```c
void game_update(void) {
    if (active_scene == SCENE_A) {
        //...
    } else if (active_scene == SCENE_B) {
        //...
    } // Maybe we have up to 5 scenes...
}
// The same structure above is also used in
`game_draw`, `game_change_scene`, and various events
```

Multiple Scenes

- We can ignore the existence of other scenes and see each scene as independent `IScene` class: more encapsulation.

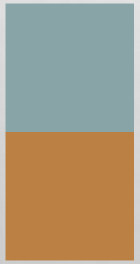
```cpp
class SceneA final : public Engine::IScene {
public:
    explicit SceneA() = default;
    void Initialize() override;
    void Terminate() override;
    void Update() override;
    void Draw() const override;
};
```

Outline

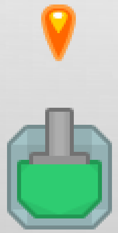
- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Controls & Objects

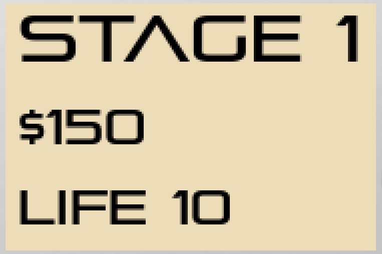
- Static images
- Images that can move, rotate, …
- Buttons
- Label (Text)



Image


Sprite

STAGE 2
STAGE 2
ImageButton


STAGE 1
$150
LIFE 10

Label (Text)

Objects & Sprites

- A simple sprite requires too much code.

```
void draw_movable_object(MovableObject obj) {
    if (obj.hidden) return;
    al_draw_bitmap(obj.img, round(obj.x - obj.w / 2),
        round(obj.y - obj.h / 2), 0);
}
void game_update() {
    for (i = 0; i < MAX_OBJ; i++) {
        if (objs[i].hidden) continue;
        objs[i].x += objs[i].vx;
        objs[i].y += objs[i].vy;
    }
}
```

Objects & Sprites

- We can define a class and specify some behaviors of the objects. Then, we can add and forget about it: one-liner for every object.

```
void SceneA::Shoot(int x, int y) {
    AddNewObject(new Bullet(x, y));
}
```

Outline

- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Objects & Controls

- A simple button requires too much code.

```
void on_mouse_down(int btn, int x, int y) {
    if (btn == 1 && pnt_in_rect(x, y, btnX, btnY, btnW, btnH)) {
        // Button clicked.
    }
}
void game_draw() {
    if (pnt_in_rect(mouse_x, mouse_y, btnX, btnY, btnW, btnH))
        al_draw_bitmap(img_btn_in, btnX, btnY, btnW, btnH);
    else
        al_draw_bitmap(img_btn_out, btnX, btnY, btnW, btnH);
}
```

Objects & Controls

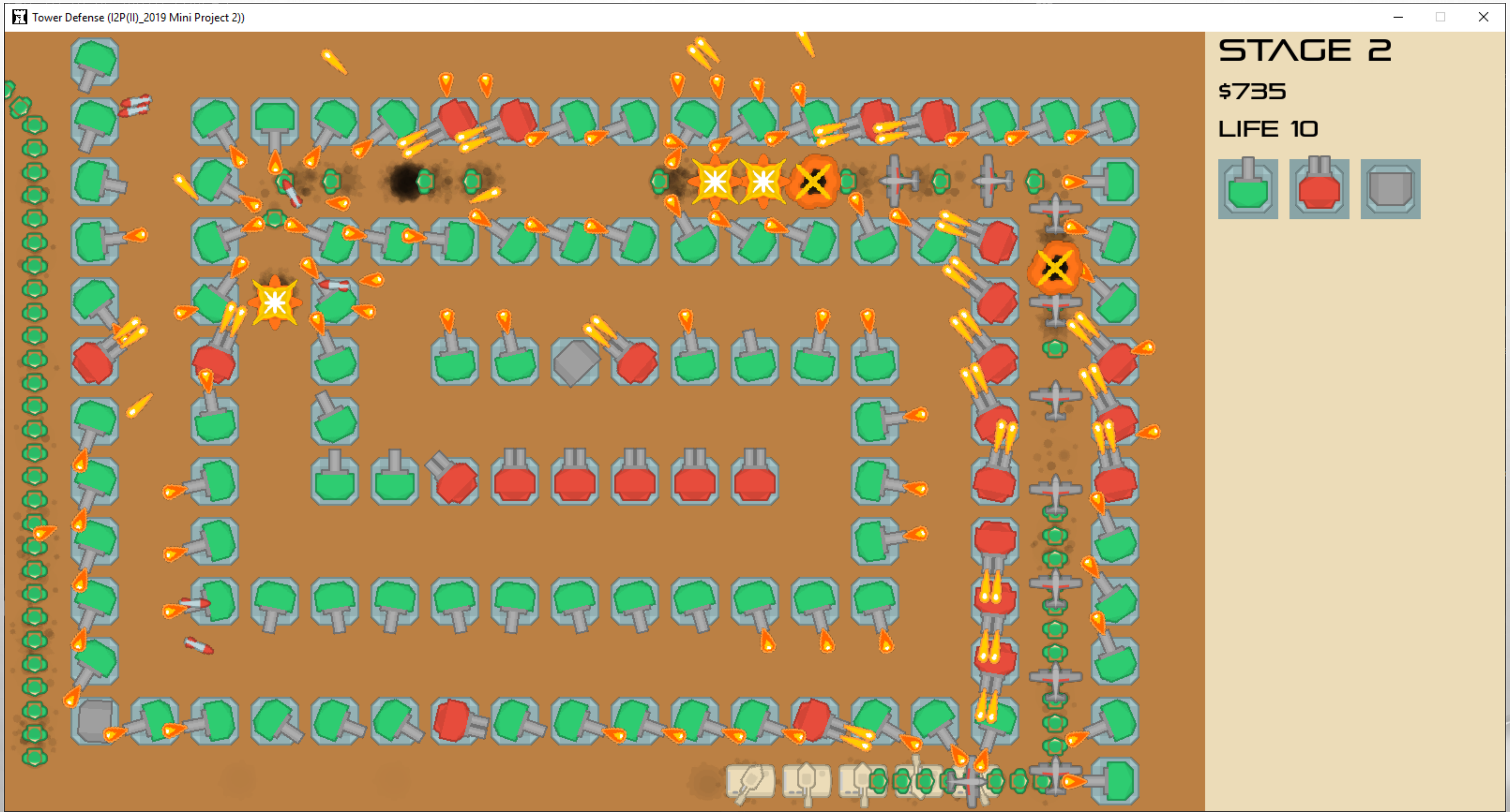
- We can ignore the drawing and mouse-in detection. For buttons, we only want to know when it is clicked. Declaring a variable just for the button is also unnecessary: higher abstraction.

```
void SceneA::BtnOnClick() { // Button clicked. }
void SceneA ::Initialize() {
    ImageButton* btn = new ImageButton("img_out.png", "img_in.png", 0, 0);
    btn->SetOnClickCallback(std::bind(&SceneA::BtnOnClick, this)));
    AddNewControlObject(btn);
}
```

Outline

- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Template Preview



Template Naming Convention

- Usually, C++ uses snake case, but we use camel case here to distinguish between STL and self-defined code.
- `std::??? (snake_case) → C++11 STL`
- `al_???, ALLEGRO_??? → Allegro5 libraries' API.`
- `Engine::??? (CamelCase) → Our own defined wrapper`
  `::??? → Classes used in game.`

Template Diagram

- Class Diagram
- Engine Class Diagram
- Engine Class Diagram Minimized
- Game Class Diagram
- Game Class Diagram Minimized
- Game Class Diagram Minimized Annotated

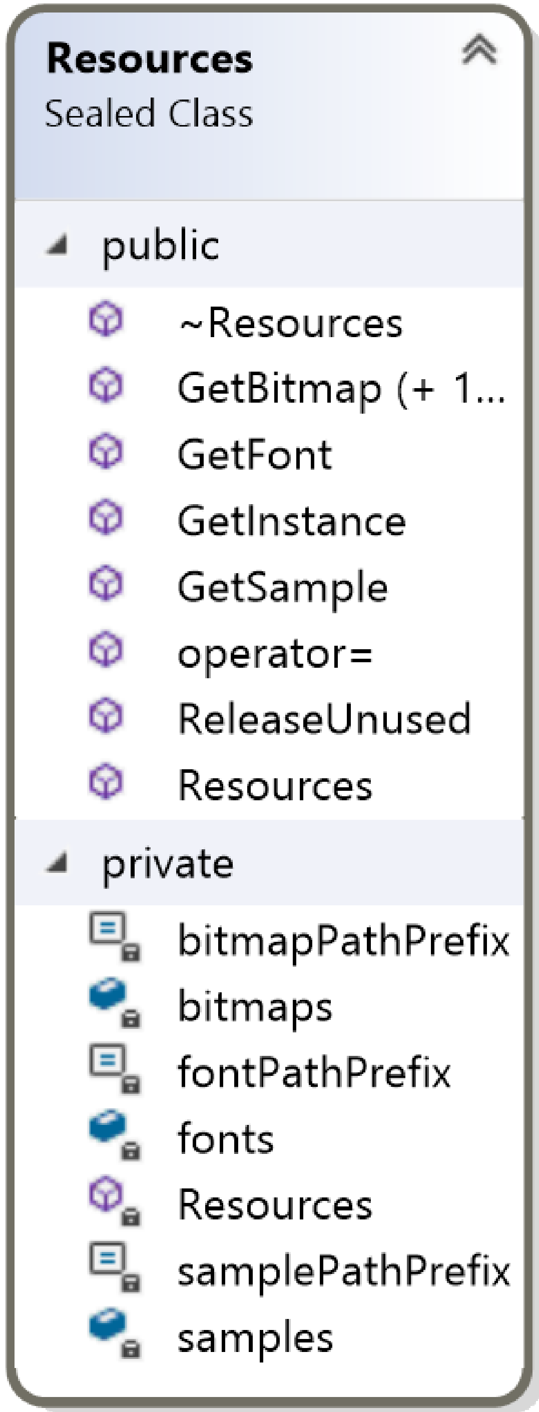
Engine code

Tower Defense

Template: Resources

`Engine::Resources`

- Abstracts all resources loading and destroy.
- Resources can be retrieved from this class directly.



Resources
Sealed Class

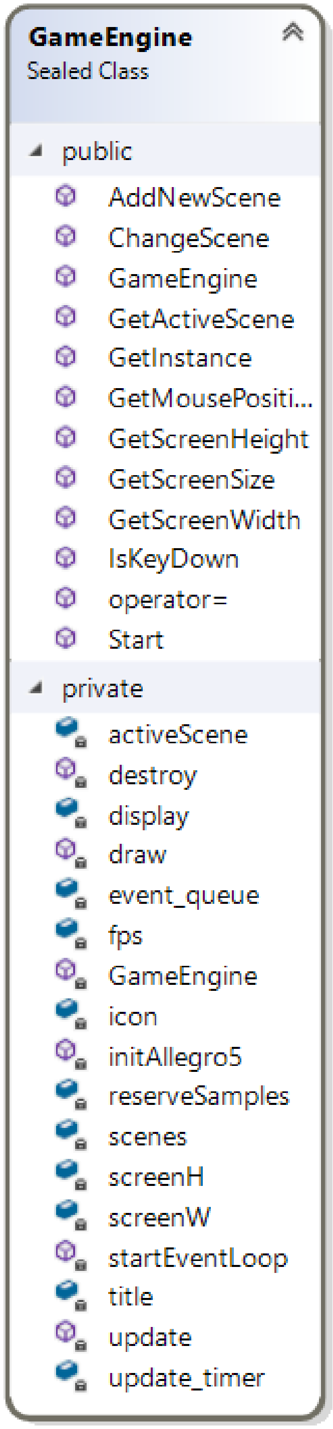
public
- ~Resources
- GetBitmap (+ 1...
- GetFont
- GetInstance
- GetSample
- operator=
- ReleaseUnused
- Resources

private
- bitmapPathPrefix
- bitmaps
- fontPathPrefix
- fonts
- Resources
- samplePathPrefix
- samples

Template: Game Engine

```
Engine::GameEngine
```

- Abstracts the entire message loop
- Manages current scene and scene changes.


GameEngine
Sealed Class

public
- AddNewScene
- ChangeScene
- GameEngine
- GetActiveScene
- GetInstance
- GetMousePositi...
- GetScreenHeight
- GetScreenSize
- GetScreenWidth
- IsKeyDown
- operator=
- Start

private
- activeScene
- destroy
- display
- draw
- event_queue
- fps
- GameEngine
- icon
- initAllegro5
- reserveSamples
- scenes
- screenH
- screenW
- startEventLoop
- title
- update
- update_timer

Template: IScene, Group

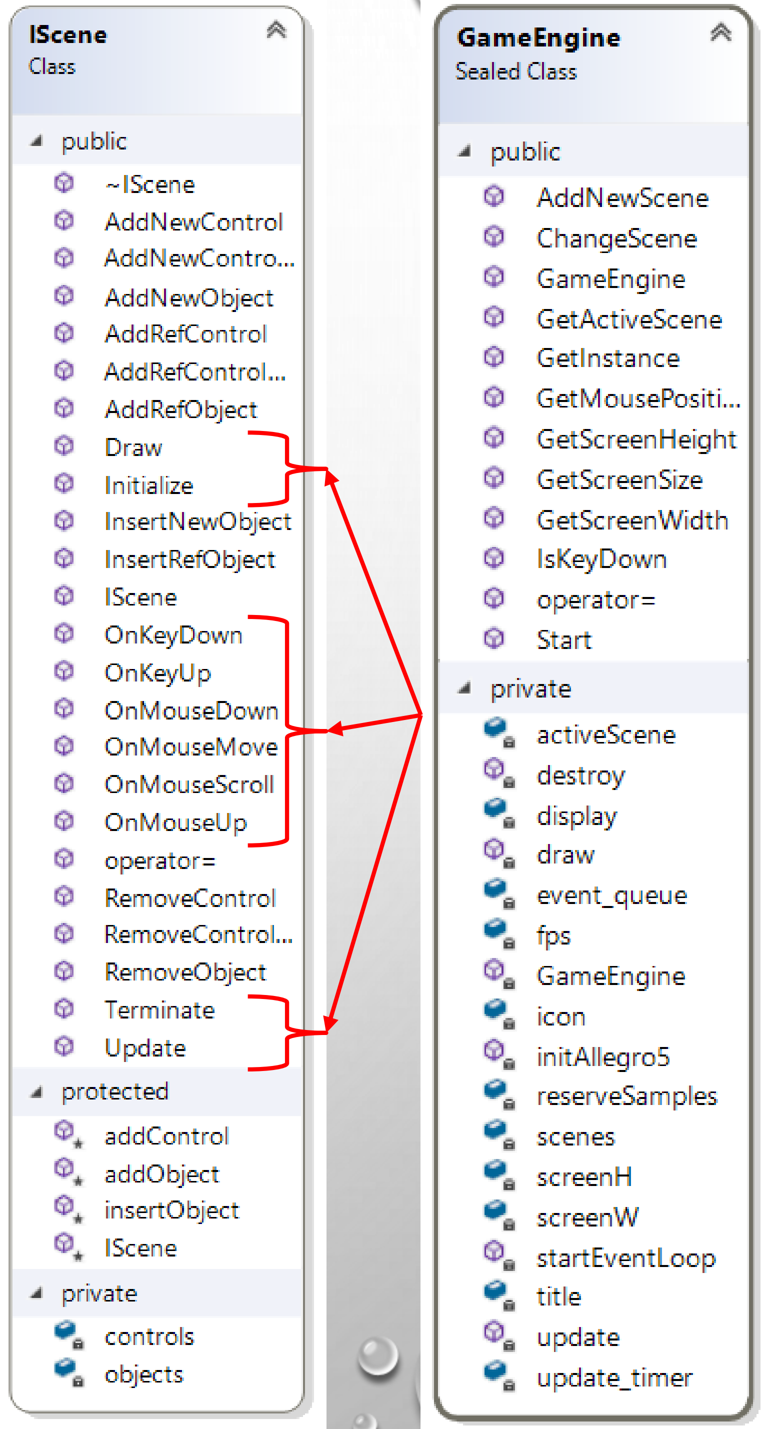
Engine::IScene

- Encapsulates a scene, must be inherited and customized.

Engine::Group

- Draw and update everything for you.

Note: We combined Group and IScene in this diagram



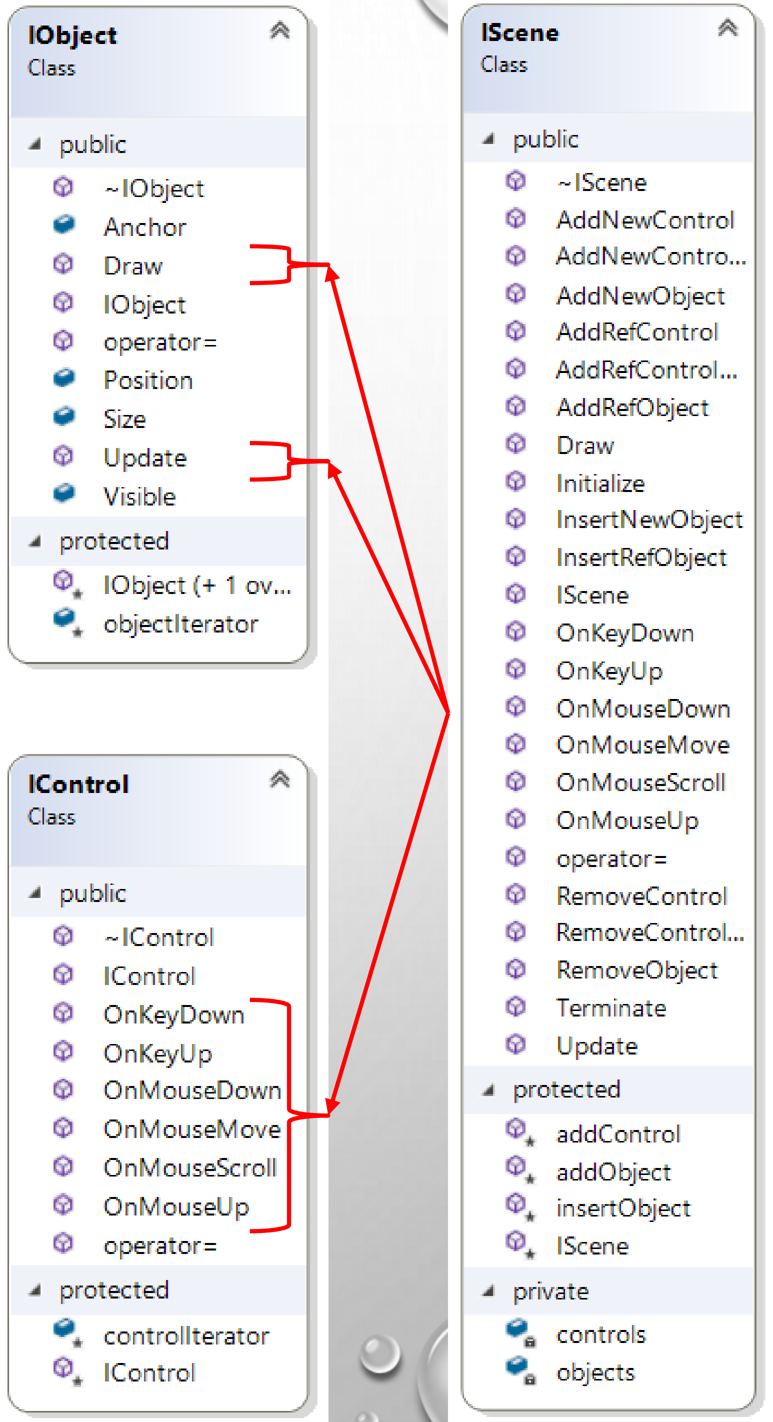
Template: IObject, IControl

Engine::IObject

- The base class of everything that can be drawn.

Engine::IControl

- The base class of everything that can receive events.



IObject
Class

public
- ~IObject
- Anchor
- Draw
- IObject
- operator=
- Position
- Size
- Update
- Visible

protected
- IObject (+ 1 ov...
- objectIterator

IControl
Class

public
- ~IControl
- IControl
- OnKeyDown
- OnKeyUp
- OnMouseDown
- OnMouseMove
- OnMouseScroll
- OnMouseUp
- operator=

protected
- controlIterator
- IControl

IScene
Class

public
- ~IScene
- AddNewControl
- AddNewContro...
- AddNewObject
- AddRefControl
- AddRefControl...
- AddRefObject
- Draw
- Initialize
- InsertNewObject
- InsertRefObject
- IScene
- OnKeyDown
- OnKeyUp
- OnMouseDown
- OnMouseMove
- OnMouseScroll
- OnMouseUp
- operator=
- RemoveControl
- RemoveControl...
- RemoveObject
- Terminate
- Update

protected
- addControl
- addObject
- insertObject
- IScene

private
- controls
- objects

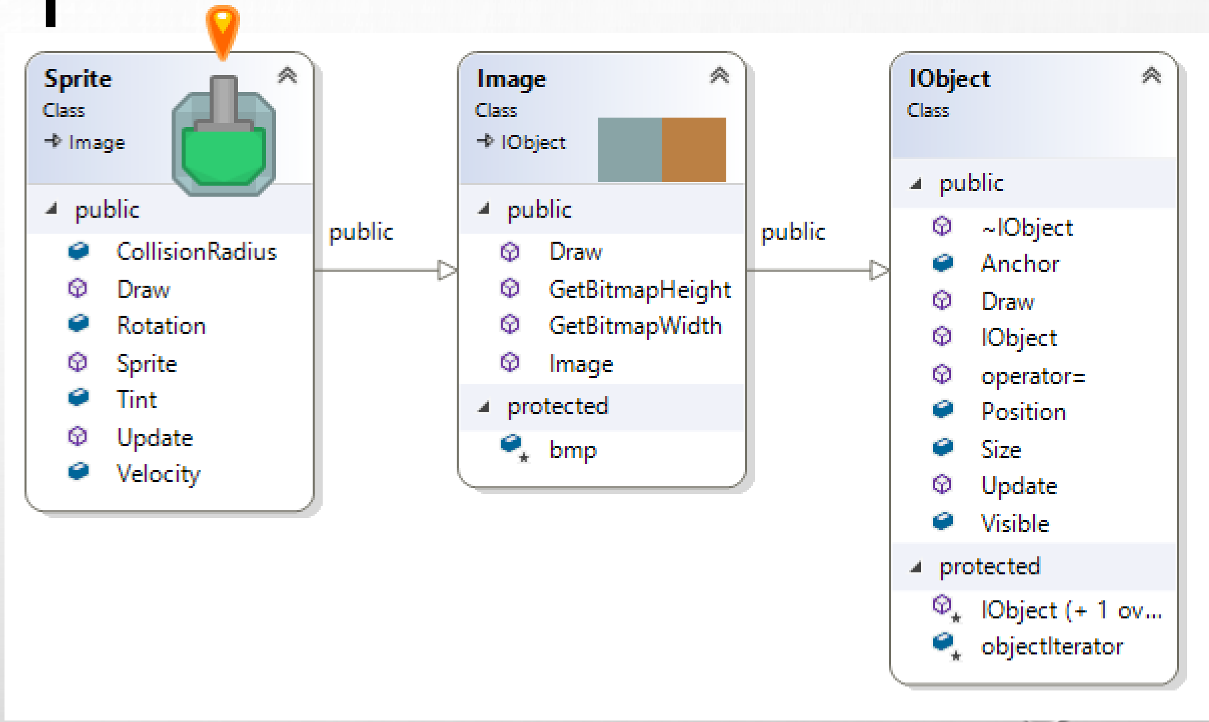
Template: Image, Sprite

```
Engine::Image :
    public Engine::IObject
```

- A simple static image object.

```
Engine::Sprite :
    public Engine::Image
```

- Supports rotation, velocity, tint, and collision radius.



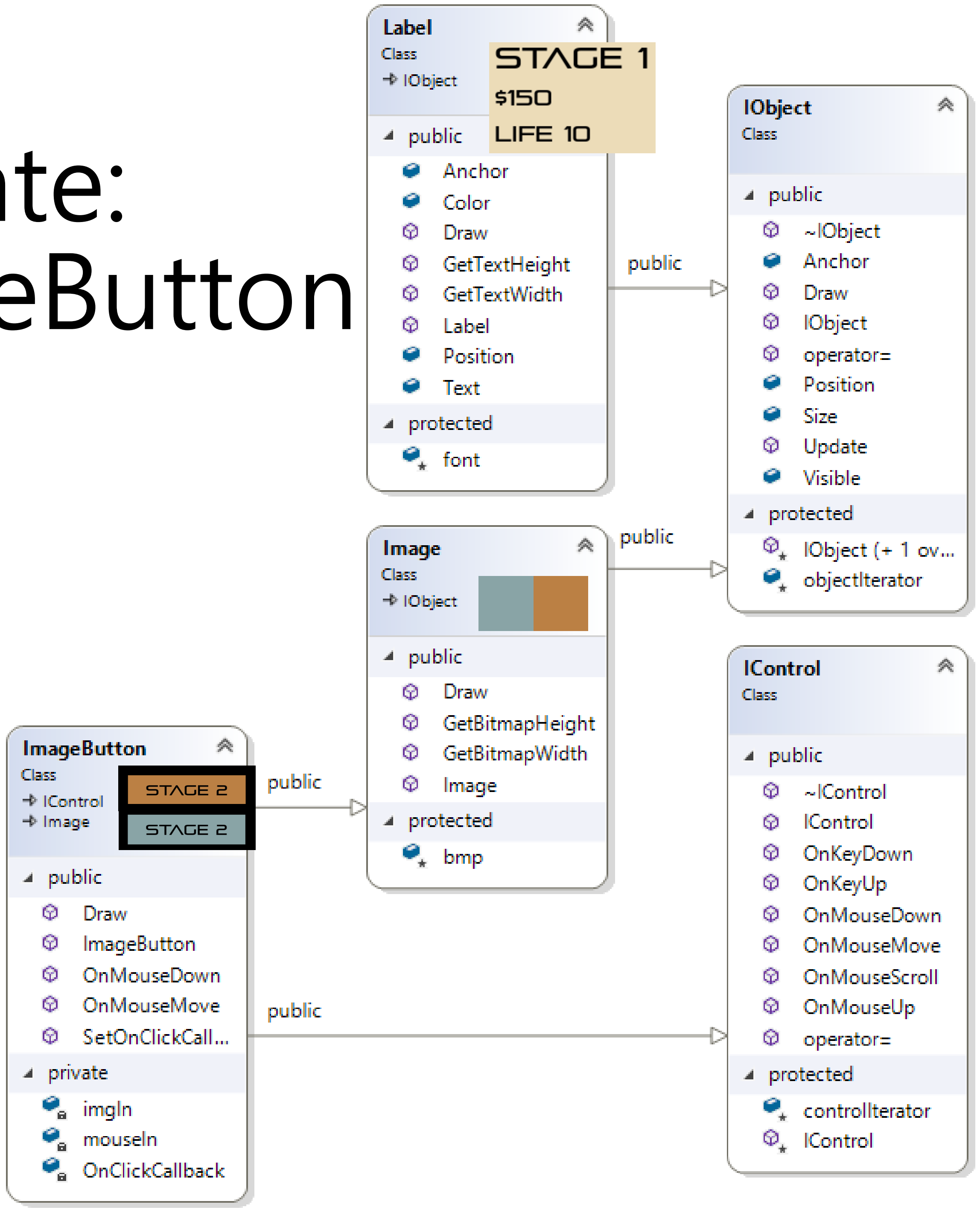
Template: Label, ImageButton

```
Engine::Label :
    public Engine::IObject
```

- A simple static text object.

```
Engine::ImageButton :
    public Engine::IObject,
    public Engine::IControl
```

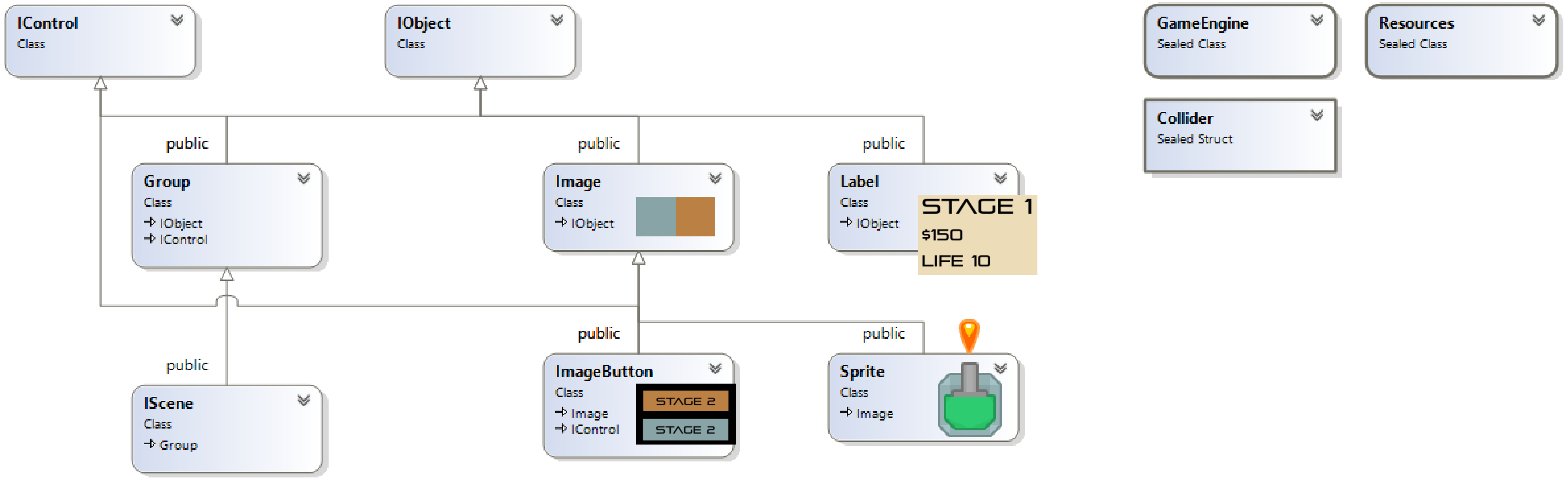
- A clickable button, changes image when mouse move.



Engine Diagram (Minimized)

(OnMouseMove, OnMouseDown, …)

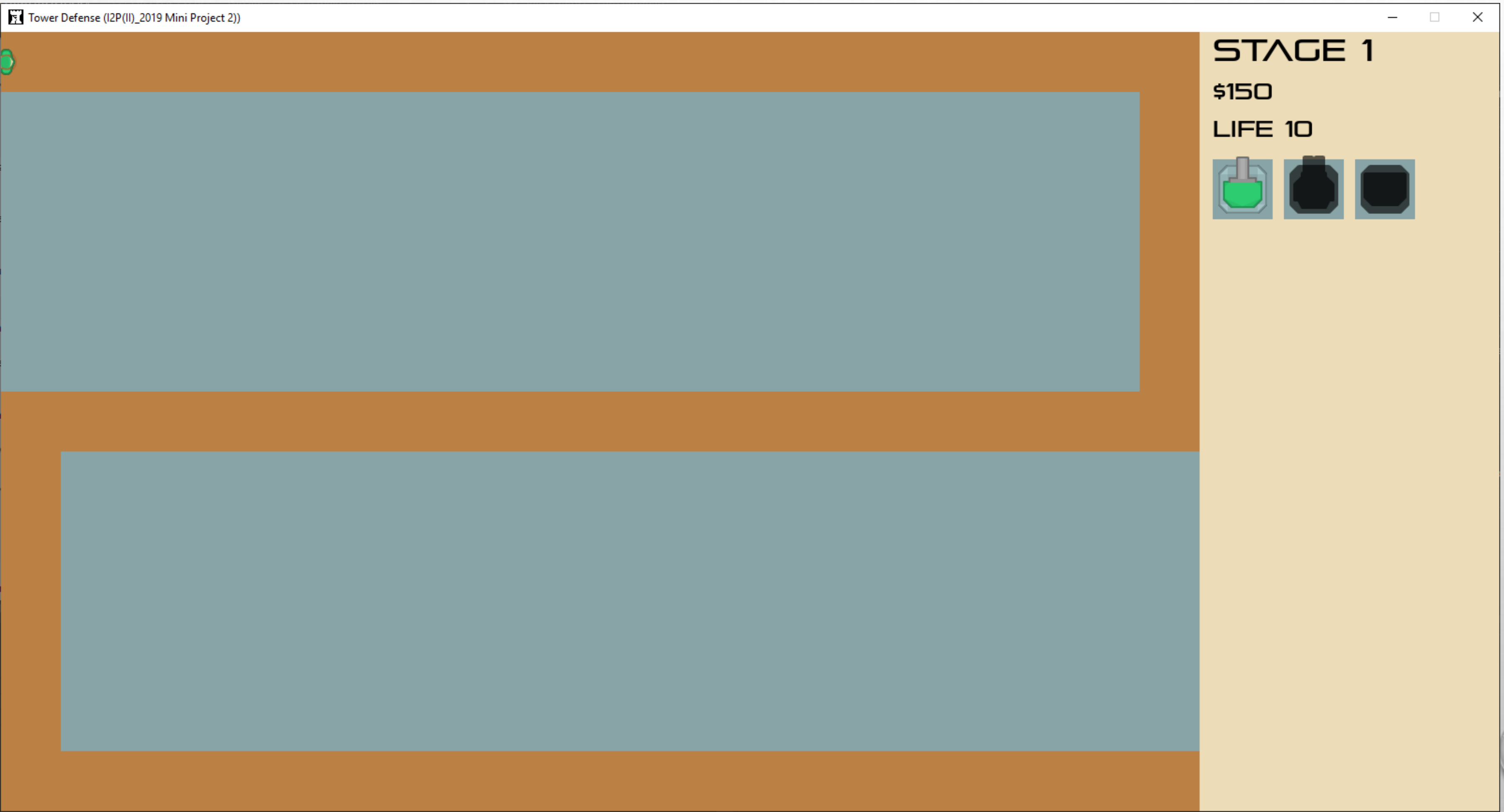
(Update, Draw)



Game code

Tower Defense

Map file format



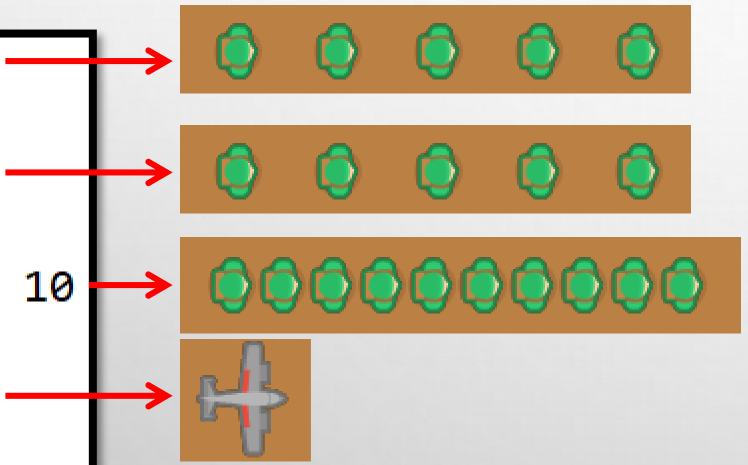
```
00000000000000000000
11111111111111111110
11111111111111111110
11111111111111111110
11111111111111111110
11111111111111111110
00000000000000000000
01111111111111111111
01111111111111111111
01111111111111111111
01111111111111111111
01111111111111111111
00000000000000000000
```

resources/map1.txt

Enemy file format

- EnemyType TimeDelayBetween Count

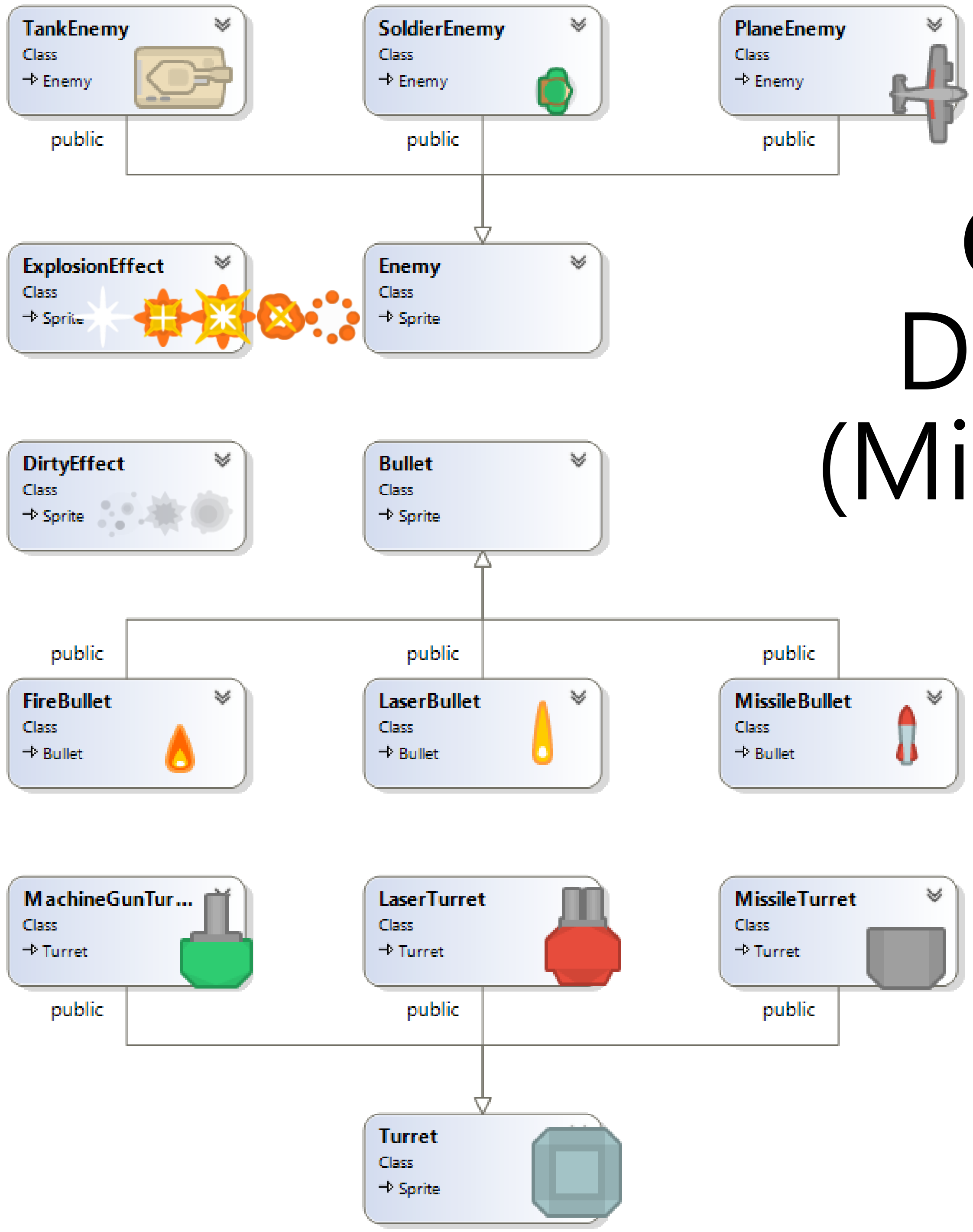
```
1 1 5
0 2 1
1 1 5
0 2 1
1 0.5 10
0 6 1
2 1 1
0 2 1
1 0.5 20
0 12 1
2 1 5
0 2 1
1 0.5 20
```

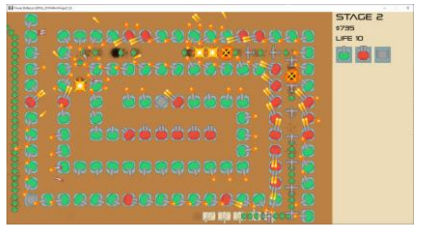


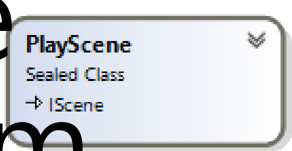
You should edit this file after adding new enemy

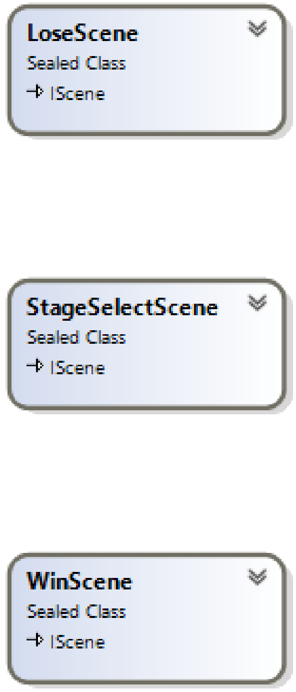
resources/enemy1.txt

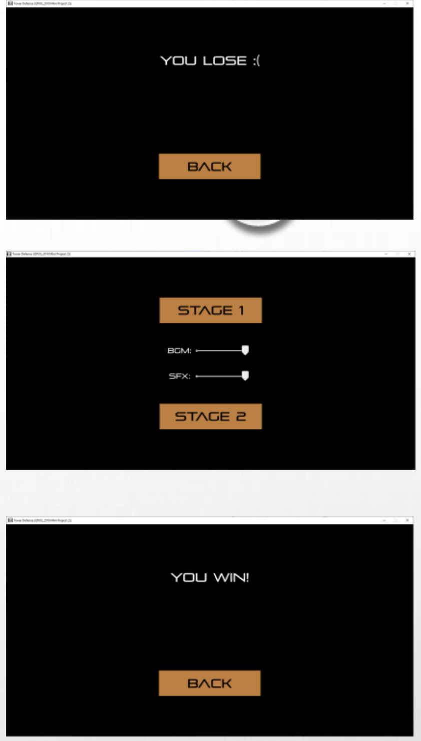
Game Diagram (Minimized)











Future of game programming

- Component system
- Physics engine
- Functional programming
- Entity component system (ECS)

- However, OOP is still a concept that cannot be abandoned in most programs that needs objects.

Outline

- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Goal

- Add starting scene and start button. (1%)
- Add 1 new tower, 1 new enemy. (1%)
- Enemy path finding. (BFS) (1%)
- Add volume control slider. (1%)
- Fix WinScene bug, find the cheat hidden in the game. (1%)
- Bonus: Continuous stages, and more… (at most +1%)

Grading Policy (1/5)

- Add starting scene and start button. (1%)
  - Button will change image when mouse enter / leave.
  - Can switch to other scene when button clicked.

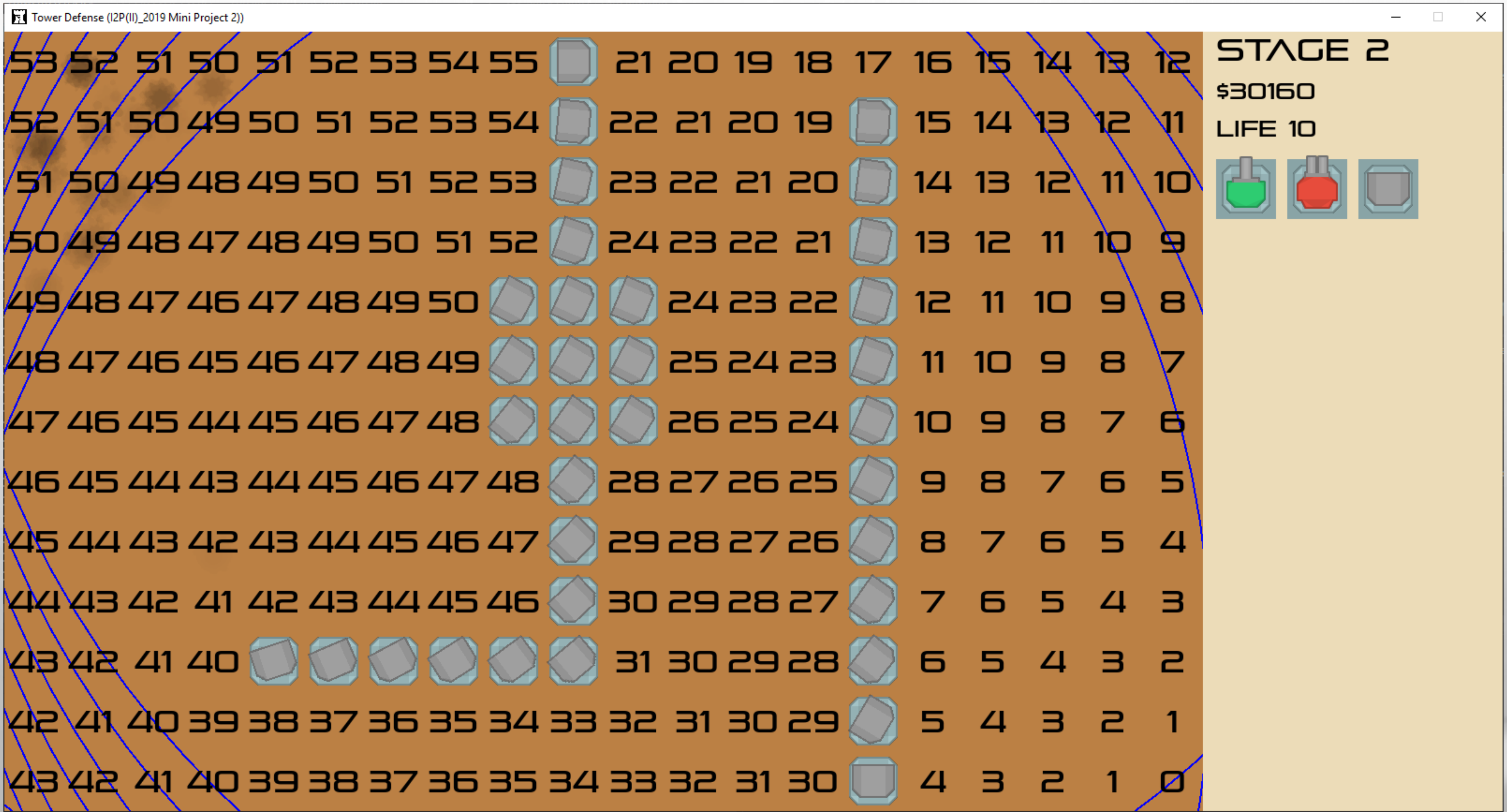
Grading Policy (2/5)

- Add 1 new tower, 1 new enemy. (1%)
  - Add 1 new tower that can be placed, and attack enemies.
  - Add 1 new enemy that can follow the path and die.
  - Both of them cannot be the same as the ones in the template. They must have different image and different behaviors.
  - Their behavior must be reasonable, if not sure, we can discuss them in iLMS.

Grading Policy (3/5)

- Enemy path finding. (BFS) (1%)
  - For stage 4, the enemy requires path finding function to move towards our base.
  - Try to implement a simple BFS counting distances between any block and our base.
  - Press **TAB** can launch the debug mode to debug easier.

Grading Policy (3/5)



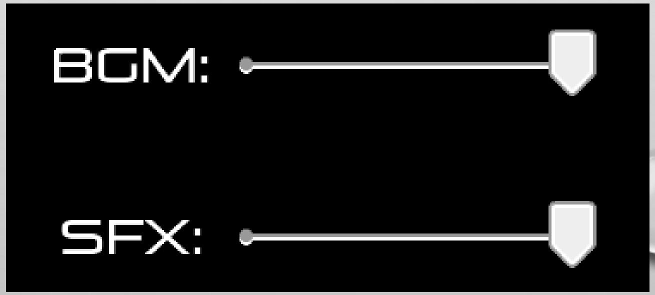
Tower Defense (I2P(II)_2019 Mini Project 2))

```
53 52 51 50 51 52 53 54 55 ▢  21 20 19 18 17 16 15 14 13 12
52 51 50 49 50 51 52 53 54 ▢  22 21 20 19 ▢  15 14 13 12 11
51 50 49 48 49 50 51 52 53 ▢  23 22 21 20 ▢  14 13 12 11 10
50 49 48 47 48 49 50 51 52 ▢  24 23 22 21 ▢  13 12 11 10 9
49 48 47 46 47 48 49 50 ▢  ▢  ▢  24 23 22 ▢  12 11 10 9  8
48 47 46 45 46 47 48 49 ▢  ▢  ▢  25 24 23 ▢  11 10 9  8  7
47 46 45 44 45 46 47 48 ▢  ▢  ▢  26 25 24 ▢  10 9  8  7  6
46 45 44 43 44 45 46 47 48 ▢  28 27 26 25 ▢  9  8  7  6  5
45 44 43 42 43 44 45 46 47 ▢  29 28 27 26 ▢  8  7  6  5  4
44 43 42 41 42 43 44 45 46 ▢  30 29 28 27 ▢  7  6  5  4  3
43 42 41 40 ▢  ▢  ▢  ▢  ▢  ▢  31 30 29 28 ▢  6  5  4  3  2
42 41 40 39 38 37 36 35 34 33 32 31 30 29 ▢  5  4  3  2  1
43 42 41 40 39 38 37 36 35 34 33 32 31 30 ▢  4  3  2  1  0
```

STAGE 2
$30160
LIFE 10

Grading Policy (4/5)

- Add volume control slider. (1%)
  - In the settings scene, we can control the sound of the music and sound effect.
  - In the game there are only mute buttons, you should implement a slider control to support easy adjustment.
  - Each of the BGM and the SFX should have their own slider.



Grading Policy (5/5)

- Fix WinScene bug, find the cheat hidden in the game. (1%)
  - The game crashes when the player wins.
  - Try to use the knowledge you learned and find out why the game crashes.
  - Make use of the tools in your IDE:
    - Stack Trace, Log, Watch variable, Breakpoint (step in / step out)
  - There is a hidden cheat code hidden in the game, you should DEMO the cheat to get the points.

Demo Date

- Date: 6/4 (Tuesday)
- Time: 19:00-21:00
- Place: 台達632
- Grade: 5%

Final Project

Final Project

Objective

- Design a PC game with Graphical User Interface (GUI) using Allegro C++ application development framework.
- You need to use OOP (Object-oriented programming) principle including class inheritance and polymorphism to develop the game.
- Try to use some ideas used in the Tower Defense Template.

Rules

- 2 students per team. If you want to work on your own, please discuss with TAs beforehand.
- Score: 10%
- Contains 2 parts:
  - Proposal 2%
  - Final DEMO 8% (+2%)

Final Project Proposal

- The final result should at least finish 50% of the proposal.
- Use C++ and Allegro5, if you want to use other libraries, ask TAs beforehand.
- Must use concepts of OOP.
- Cannot use Mini-Project 2 as your final project.

Proposal Spec.

- Deadline: 6/7 (Monday) Team up & Proposal 2%
- Format: (Sample link, English version)
  - Team number, student id and name
  - Game title / Game type (genre)
  - Details / Brief intro
  - How to play / Controls
  - High risk analysis
  - High value analysis
  - Games alike
  - Game preview
  - OOP Structure
  - Classes and description

Grading Policy

- Date: 6/24 (Monday) DEMO 8% (+2%)
- Basic
  - 30% At least finish 50% of your proposal
  - 20% The main mechanics of your game
  - 10% Starting scene
  - 10% Use images
  - 10% Music and Sound Effects
  - 10% Keyboard & Mouse

Grading Policy: Functional Bonus

- Bonus (+1%) (Choose either 3 of below)
  - Animation, Particle effects
  - File save / load (e.g. Scoreboard, Game saves)
  - Two players / Split screen
  - Special level design
  - AI of enemy
  - Physical Engine
  - Fun, Performance, Creative.

Grading Policy: Code Bonus

- Bonus (+1%) (Choose either 3 of below)
  - Git (Version control must use merge)
  - Smart Pointers
  - Multi-thread smooth loading (pthread, std::thread, …)
  - Online Multiplayer (sockets, …)
  - Elegant Coding Style (OOP, Design Patterns, …)
  - Try-catch and Lambda function
  - Algorithms not covered in this course (A* path finding, Separate-Axis Theorem, …)

Demo

- 6/24 (Monday) DEMO 8% (+2%)
- Explain the relationships between your classes, code can be shown if essential.
- Explain your features and try to get the scores.
- Demo time spreadsheet will be announced on iLMS, you should type your name in the time slots.

Hackathon

I2P(II)_2019_SR

Today’ s Goal

- At most +1.2%
- Add starting scene and start button. (+0.4%)
- Add 1 new tower, 1 new enemy. (+0.4%)
- Enemy path finding. (BFS) (+0.4%)
- Add volume control slider. (+0.4%)
- Fix WinScene bug, find the cheat hidden in the game. (+0.4%)
- Final Proposal (+0.8%) (Must finish all details)

Q&A

You can ask any question here.