Tower Defense

Mini Project 2 Package

Before we start,

Demo Date

- Date: 6/11 (Tuesday)
- Time: 1320-1510
- Place: 資電館
- Grade: 5%

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

Outline

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

Outline

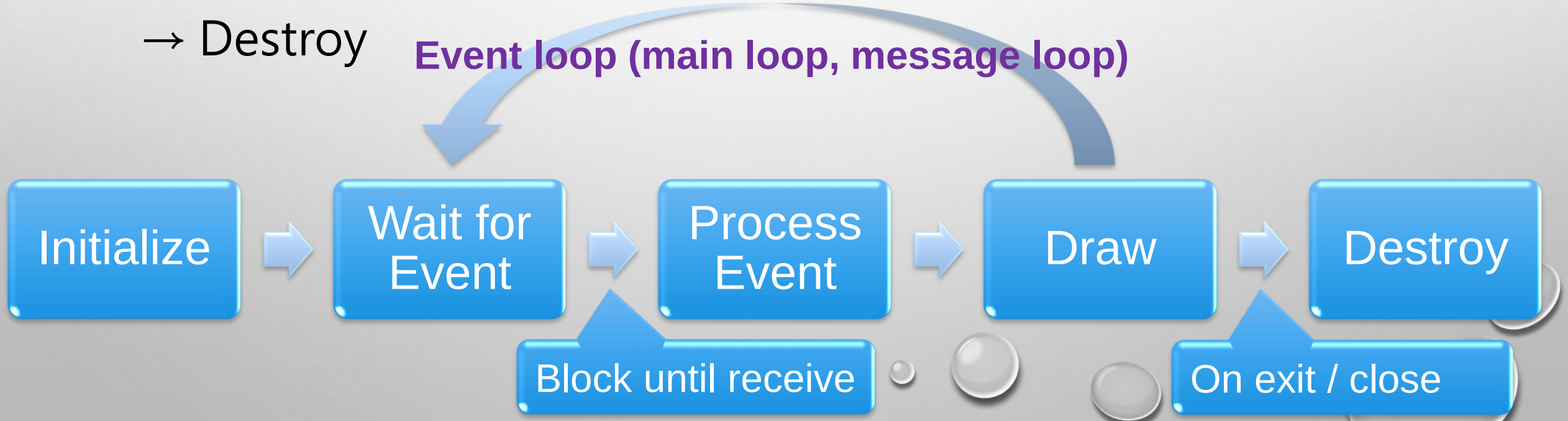
- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Allegro5

- A cross-platform library mainly aimed at video game and multimedia programming.
- Supported on Windows, Linux, Mac OSX, iPhone and Android.
- User-friendly, intuitive C API usable from C++ and many other languages.
- Hardware accelerated bitmap and graphical primitive drawing support. (via OpenGL or Direct3D)

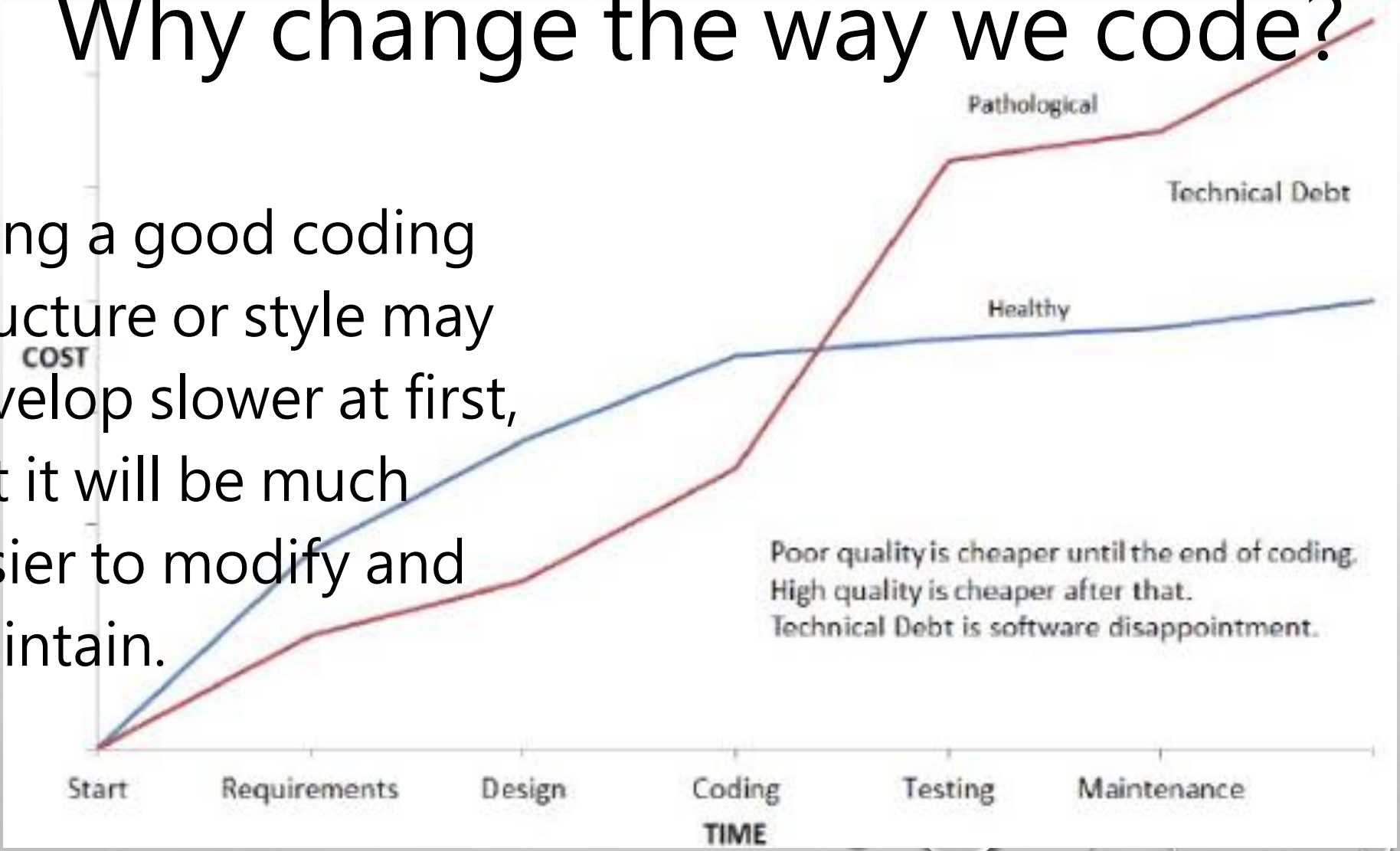
Program Flow in Allegro5

- 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.

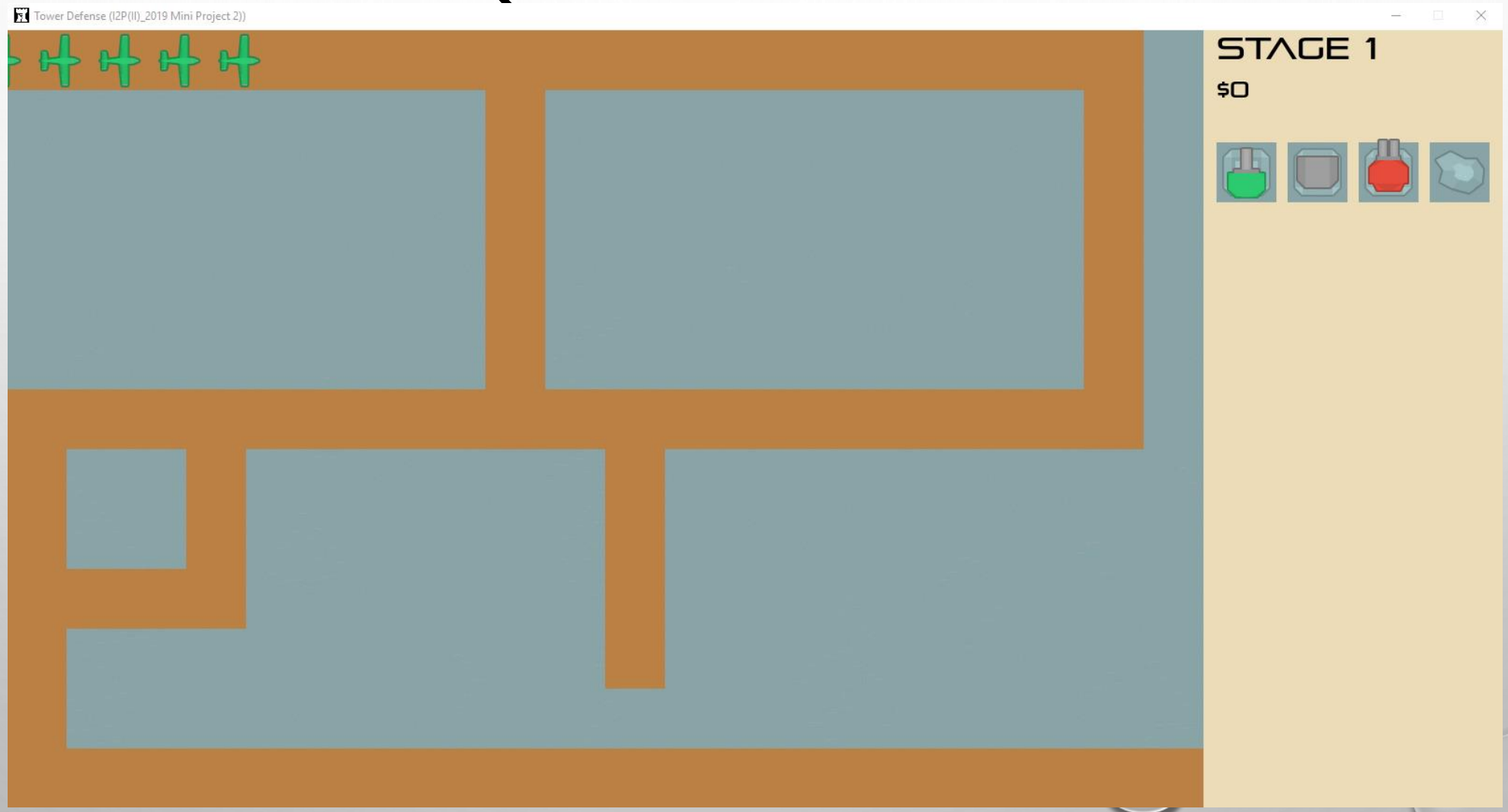


Source: http://www.critical-logic.com/services/qa-project-management/

Outline

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

Quick demo



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.

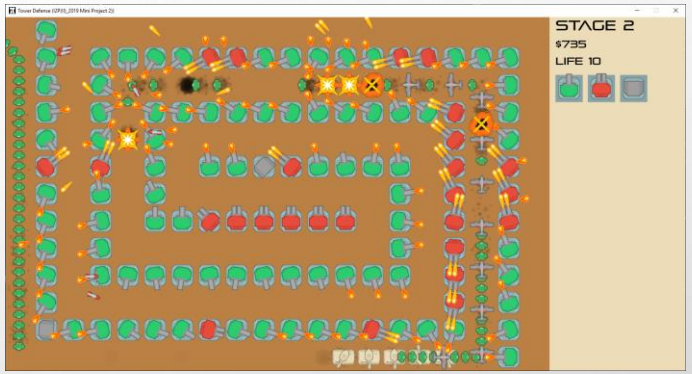
```
Resources::GetInstance().GetBitmap("img.png");
//...
// Automatically free resources.
```

Outline

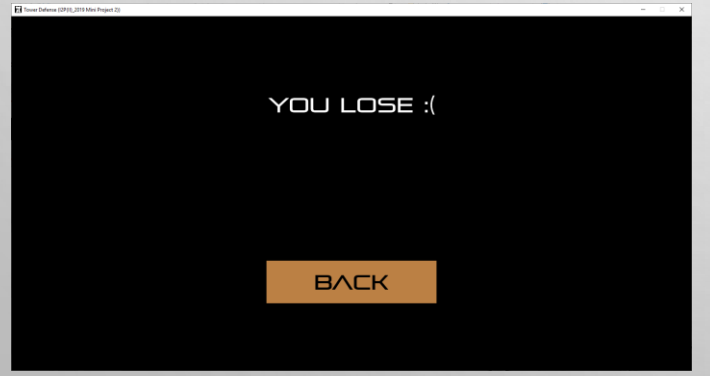
- Quick review
- Resources
- Scenes
- Objects & Sprites
- Objects & Controls
- Template & Code structure
- Goal & Grading Policy

Scenes

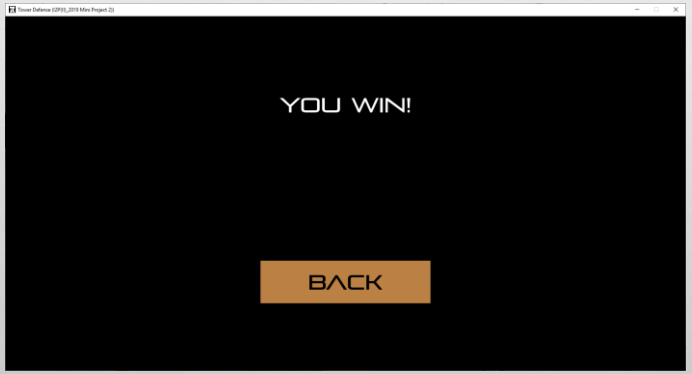
- All scenes should be independent.
- Change between scenes with only a function call.



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.

```
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.

```
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

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.

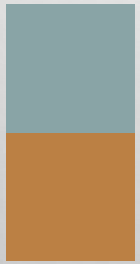
```
void SceneA::Shoot(int x, int y) {
    AddNewObject(new Bullet(x, y));
}
```

Outline

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

Controls & Objects

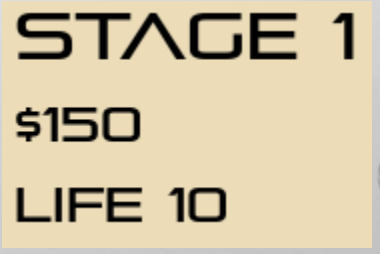
- Static images
- Images that can move, rotate, …
- Buttons
- Label (Text)



Image

Sprite

ImageButton



Label (Text)

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

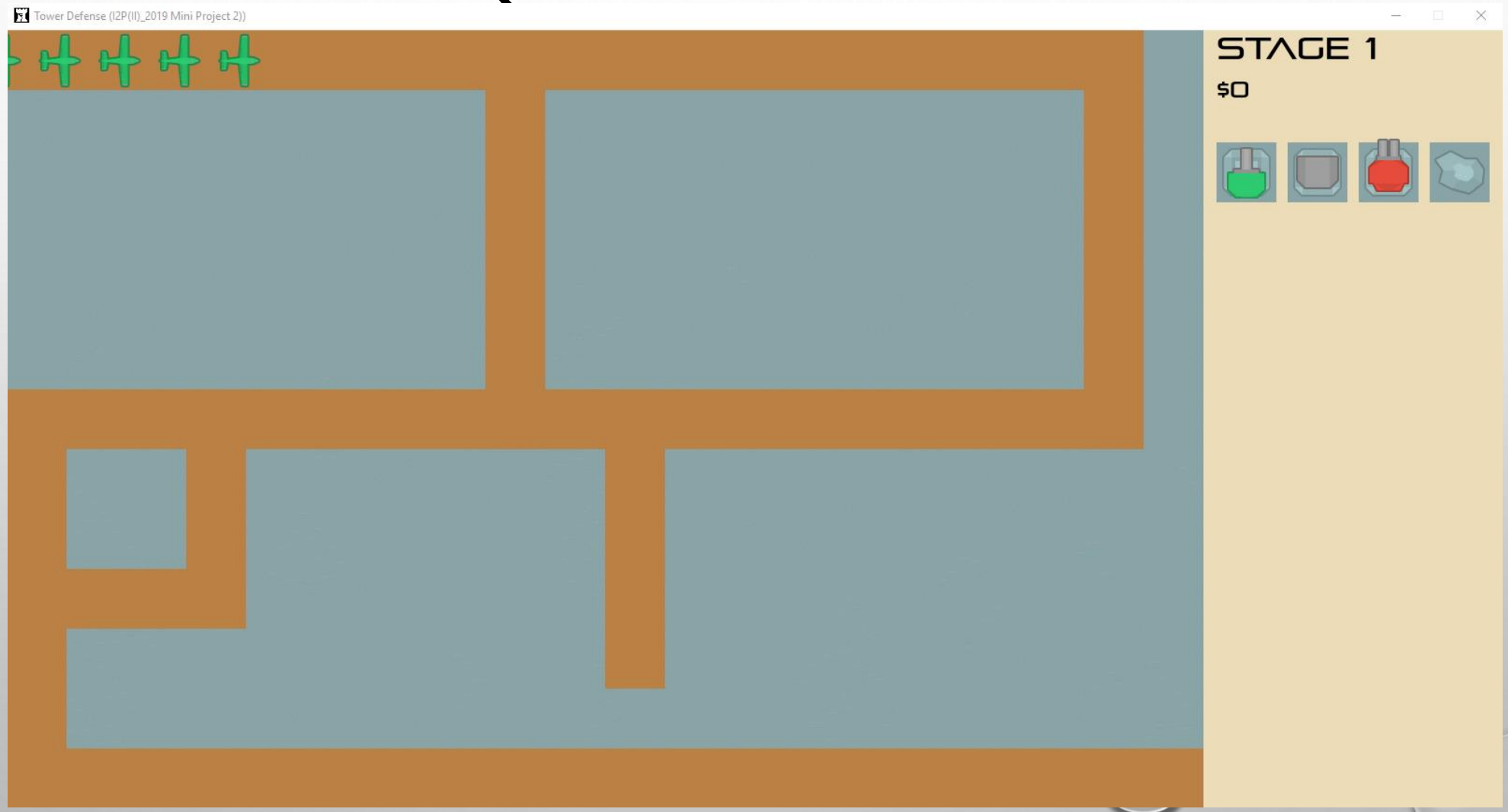
- We can ignore the drawing and mouse-in detection. For buttons, we only want to know when it is clicked. Declaring a variable just for the button is also unnecessary: higher abstraction.

```
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);
}
```

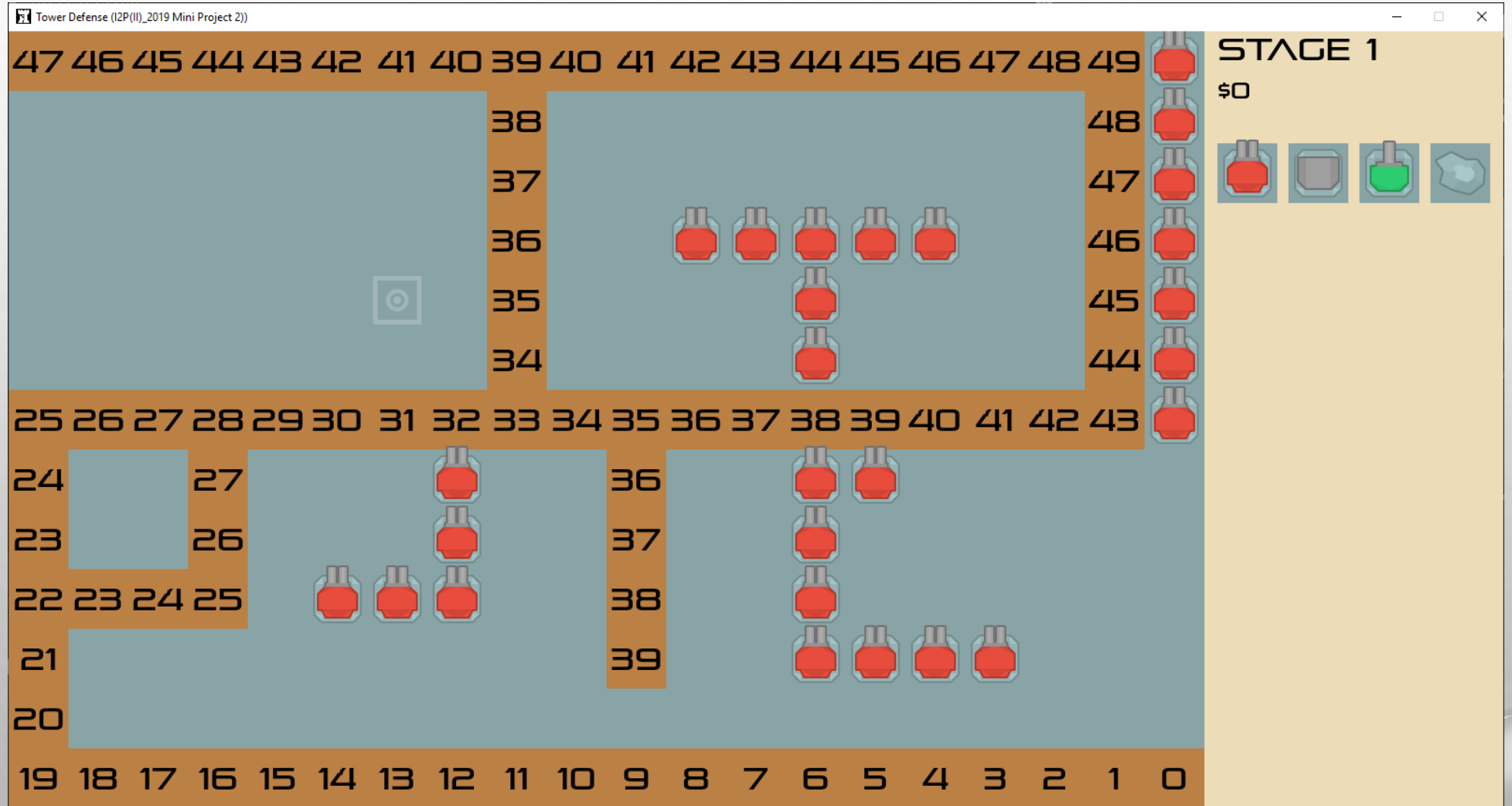
Outline

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

Quick demo



Template Preview (Debug mode)



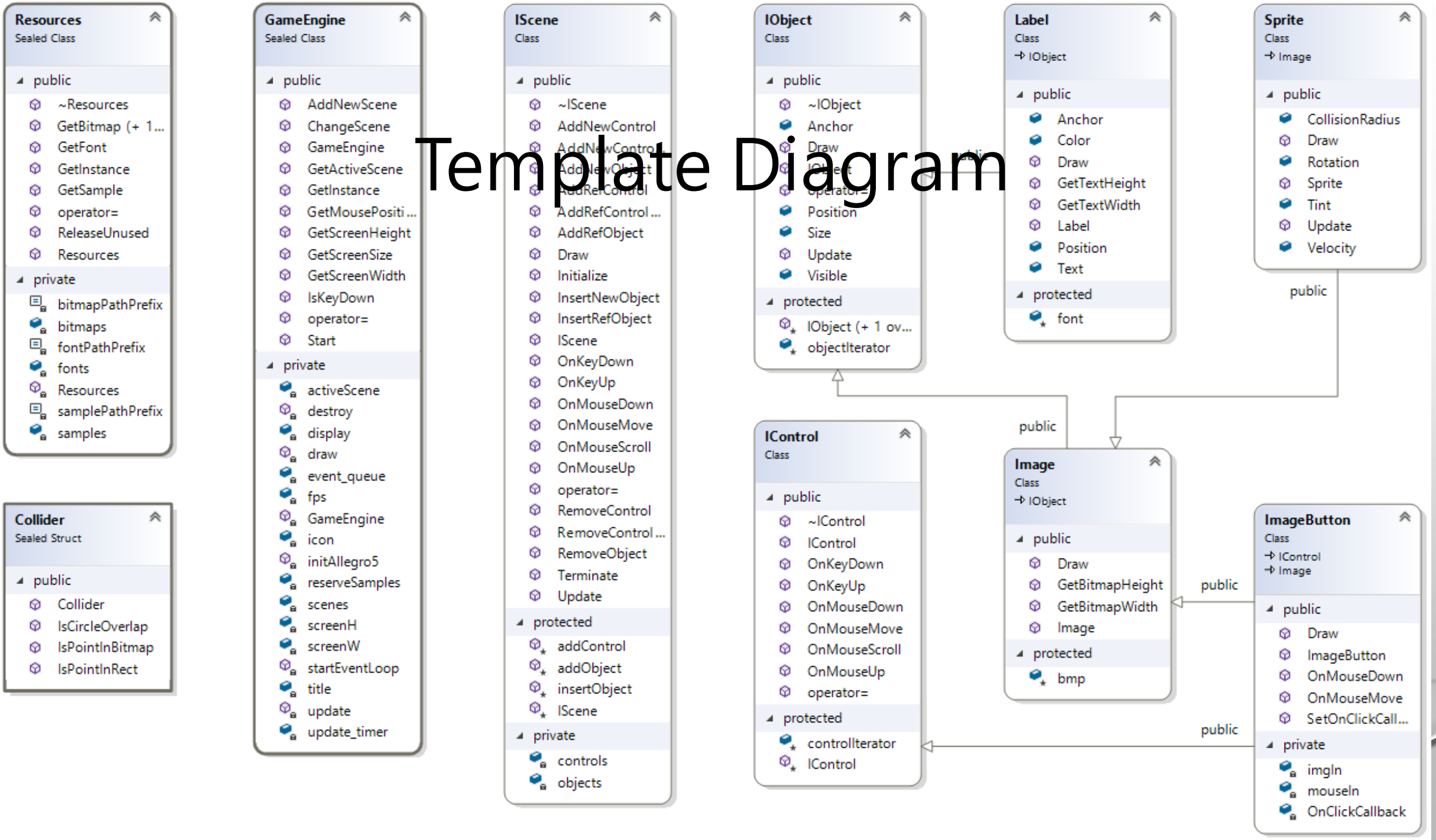
Template Naming Convention

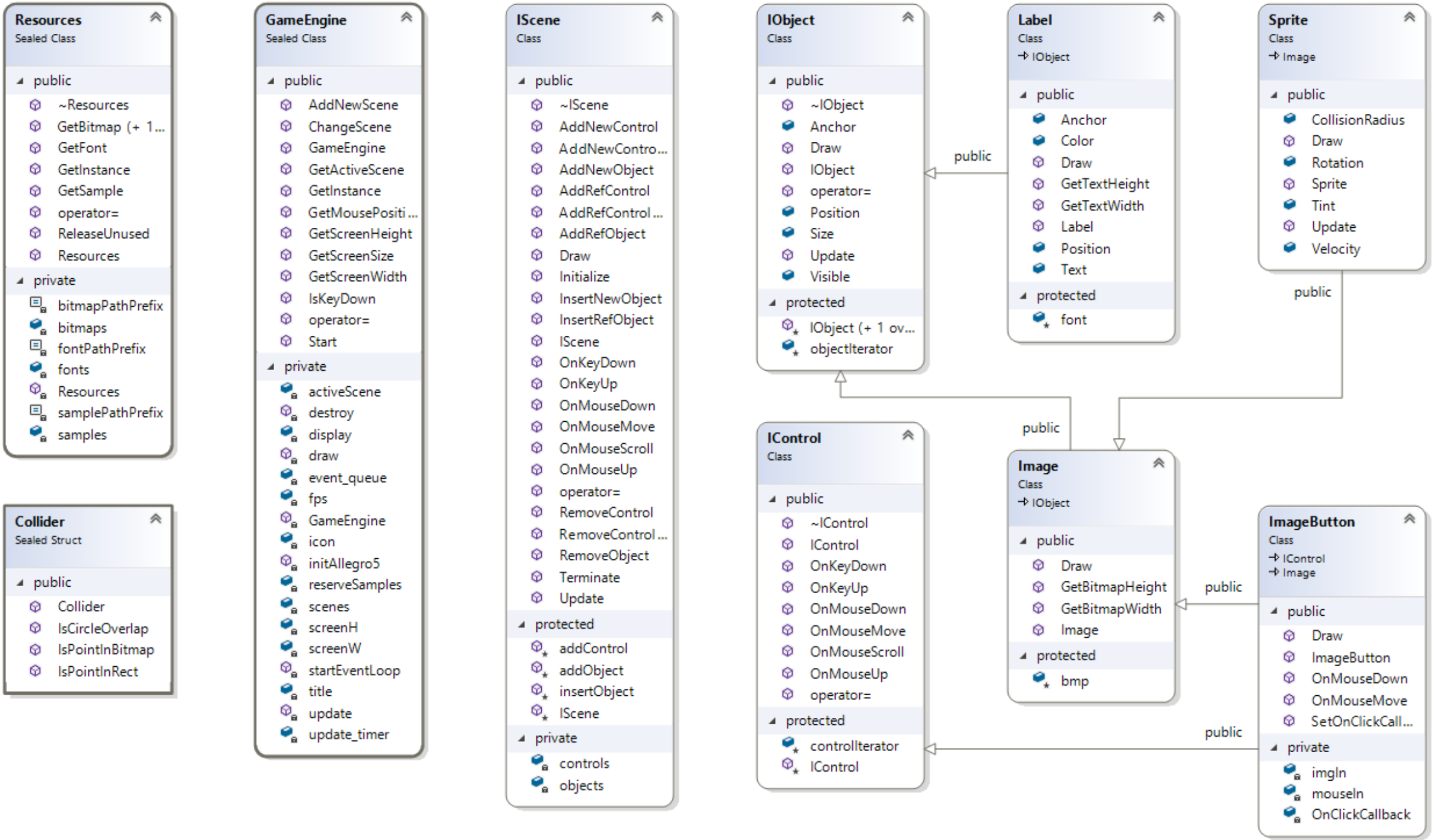
- Usually, C++ uses snake case, but we use camel case here to distinguish between STL and self-defined code.
- `std::??? (snake_case) → C++11 STL`
- `al_???, ALLEGRO_??? → Allegro5 libraries’ API.`
- `Engine::??? (CamelCase) → Our own defined wrapper ::??? → Classes used in game.`

Template Diagram

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

Template Diagram




Resources
Sealed Class
public
~Resources
GetBitmap (+ 1...
GetFont
GetInstance
GetSample
operator=
ReleaseUnused
Resources
private
bitmapPathPrefix
bitmaps
fontPathPrefix
fonts
Resources
samplePathPrefix
samples
Collider
Sealed Struct
public
Collider
IsCircleOverlap
IsPointInBitmap
IsPointInRect
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
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
IObject
Class
public
~IObject
Anchor
Draw
IObject
operator=
Position
Size
Update
Visible
protected
IObject (+ 1 ov...
objectIterator
public
Label
Class
IObject
public
Anchor
Color
Draw
GetTextHeight
GetTextWidth
Label
Position
Text
protected
font
Sprite
Class
Image
public
CollisionRadius
Draw
Rotation
Sprite
Tint
Update
Velocity
public
public
IControl
Class
public
~IControl
IControl
OnKeyDown
OnKeyUp
OnMouseDown
OnMouseMove
OnMouseScroll
OnMouseUp
operator=
protected
controlIterator
IControl
Image
Class
IObject
public
Draw
GetBitmapHeight
GetBitmapWidth
Image
protected
bmp
public
ImageButton
Class
IControl
Image
public
Draw
ImageButton
OnMouseDown
OnMouseMove
SetOnClickCall...
private
imgIn
mouseIn
OnClickCallback
public
public

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

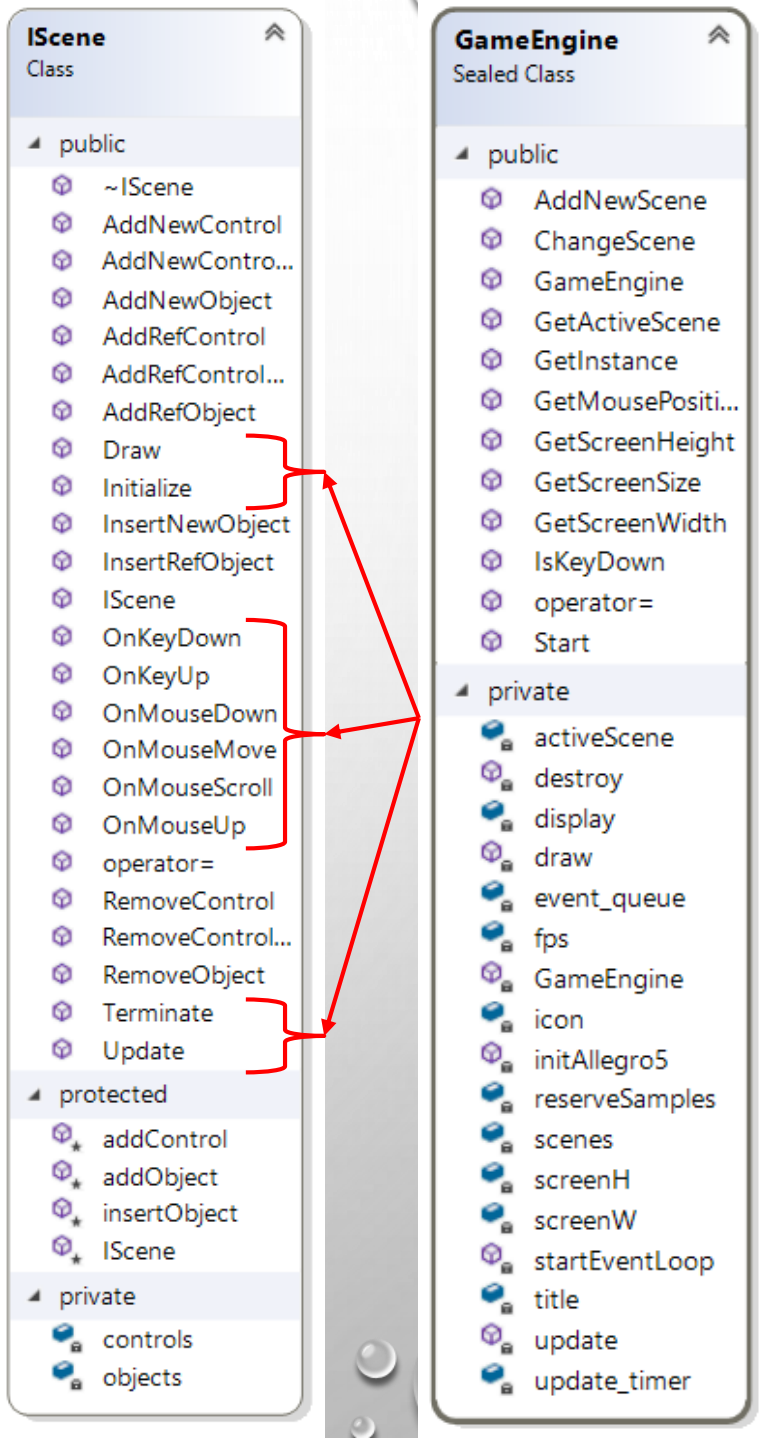
public
- AddNewScene
- ChangeScene
- GameEngine
- GetActiveScene
- GetInstance
- GetMousePositi...
- GetScreenHeight
- GetScreenSize
- GetScreenWidth
- IsKeyDown
- operator=
- Start

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

Template: IScene

`Engine::IScene`

- Encapsulates a scene, must be inherited and customized.
- Draw and update everything for you.



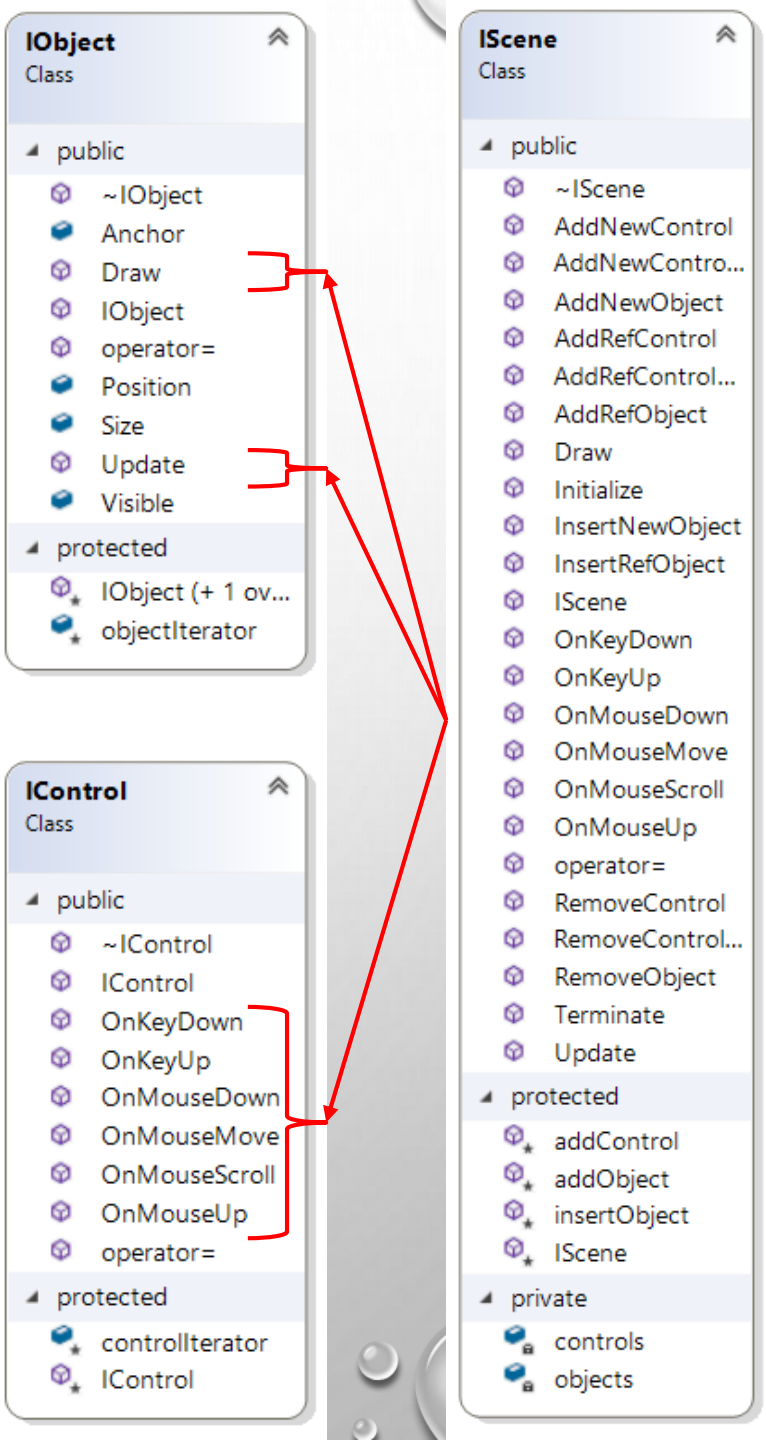
Template: IObject, IControl

`Engine::IObject`

- The base class of everything that can be drawn.

`Engine::IControl`

- The base class of everything that can receive events.



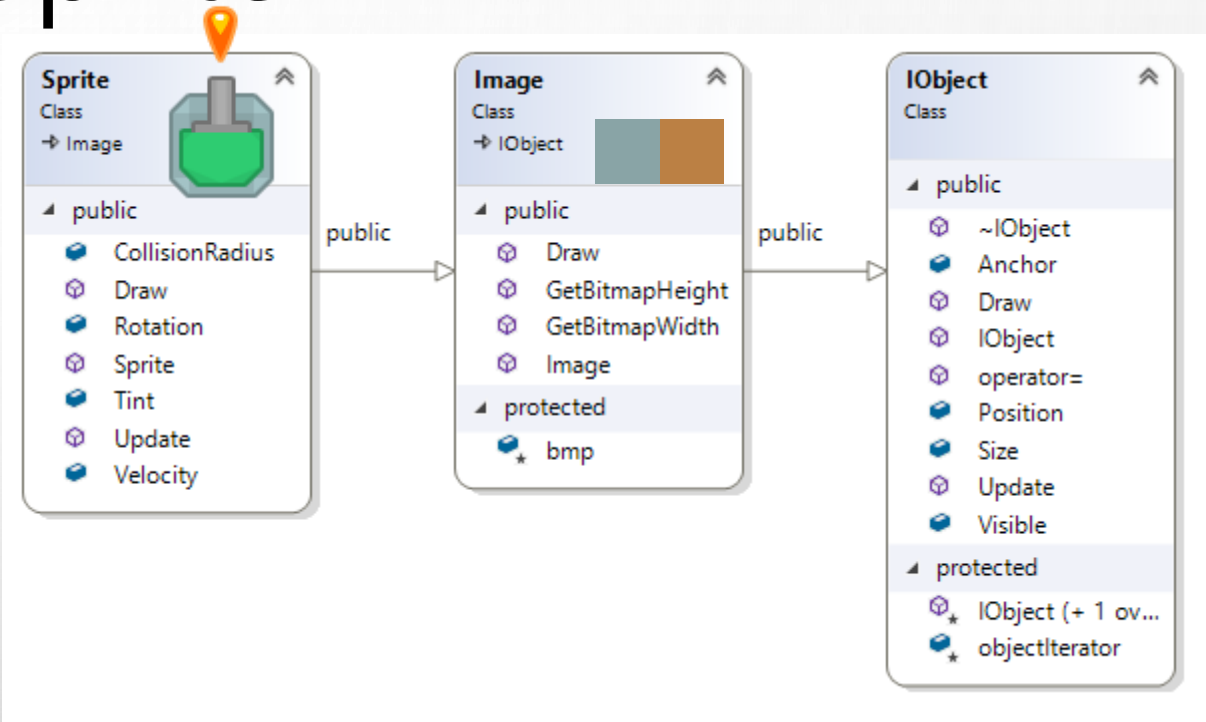
Template: Image, Sprite

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

- A simple static image object.

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

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



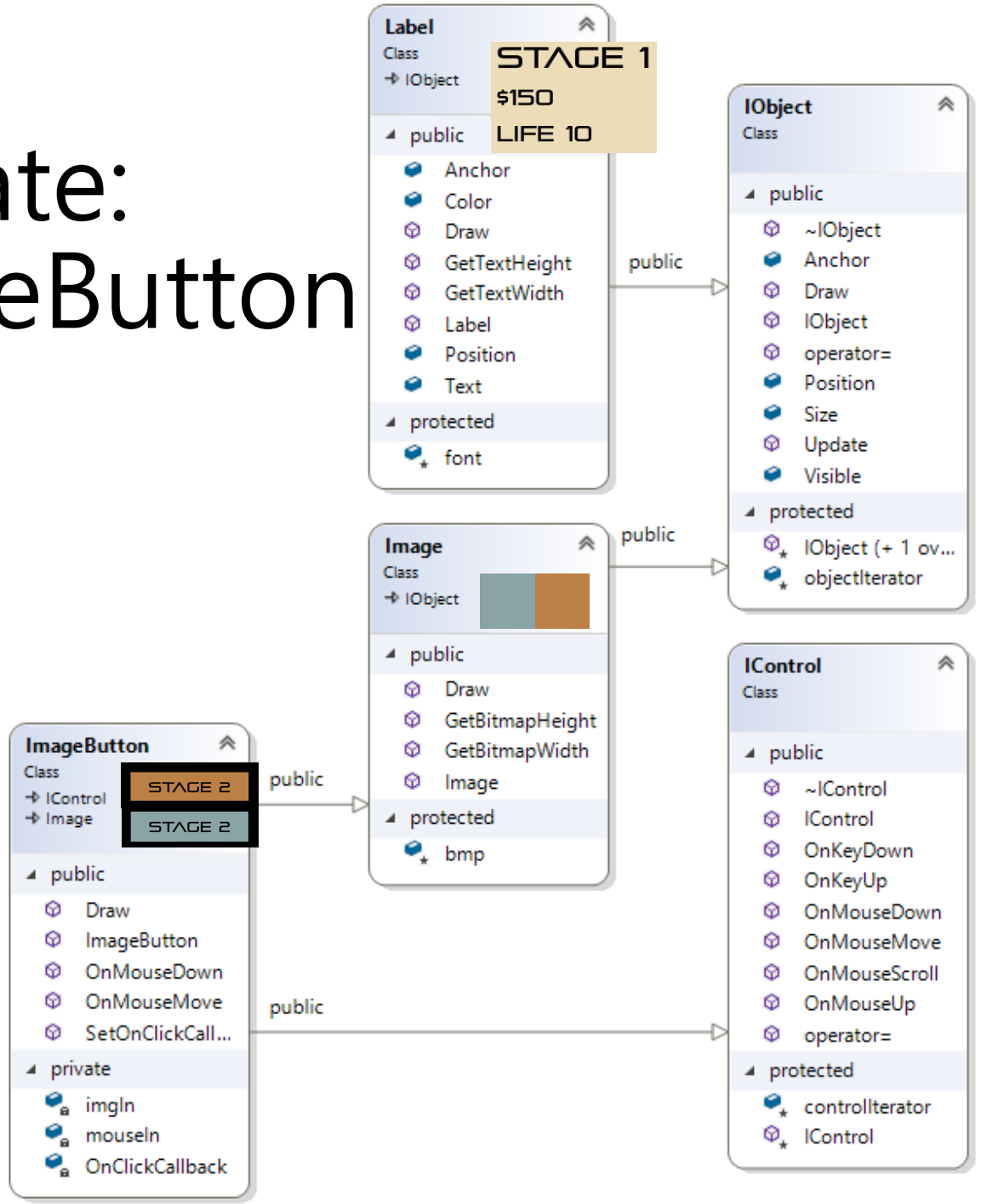
Template: Label, ImageButton

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

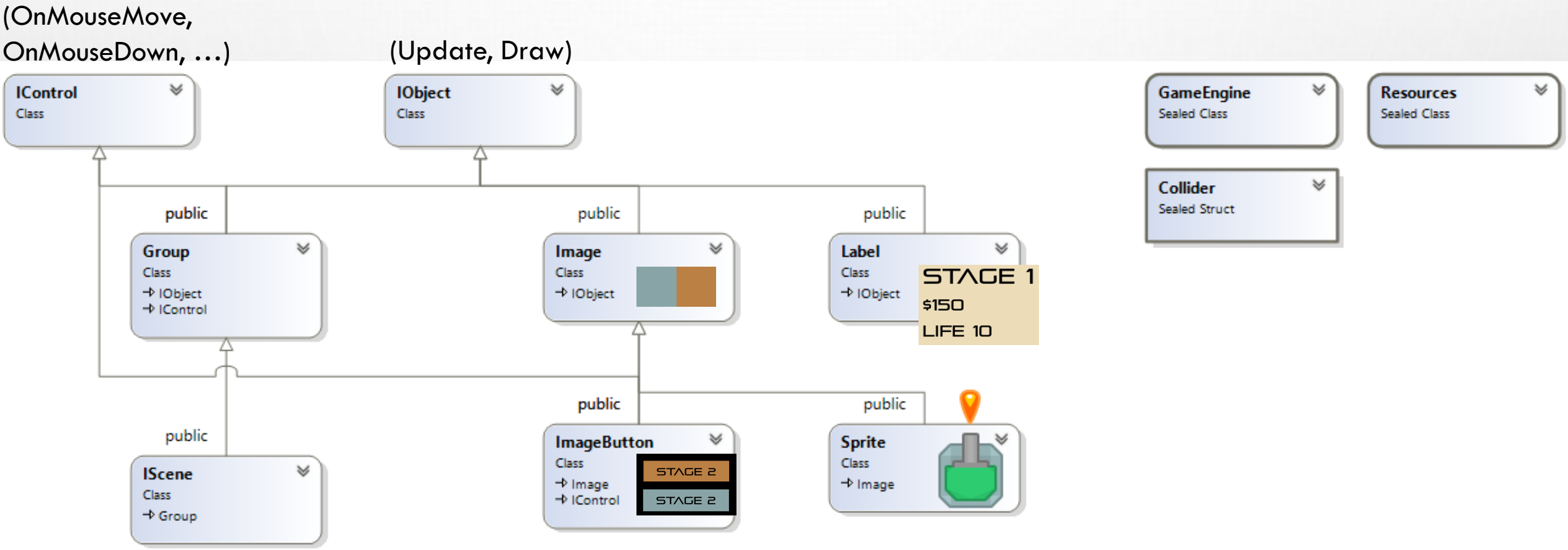
- A simple static text object.

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

- A clickable button, changes image when mouse move.



Engine Diagram (Minimized)



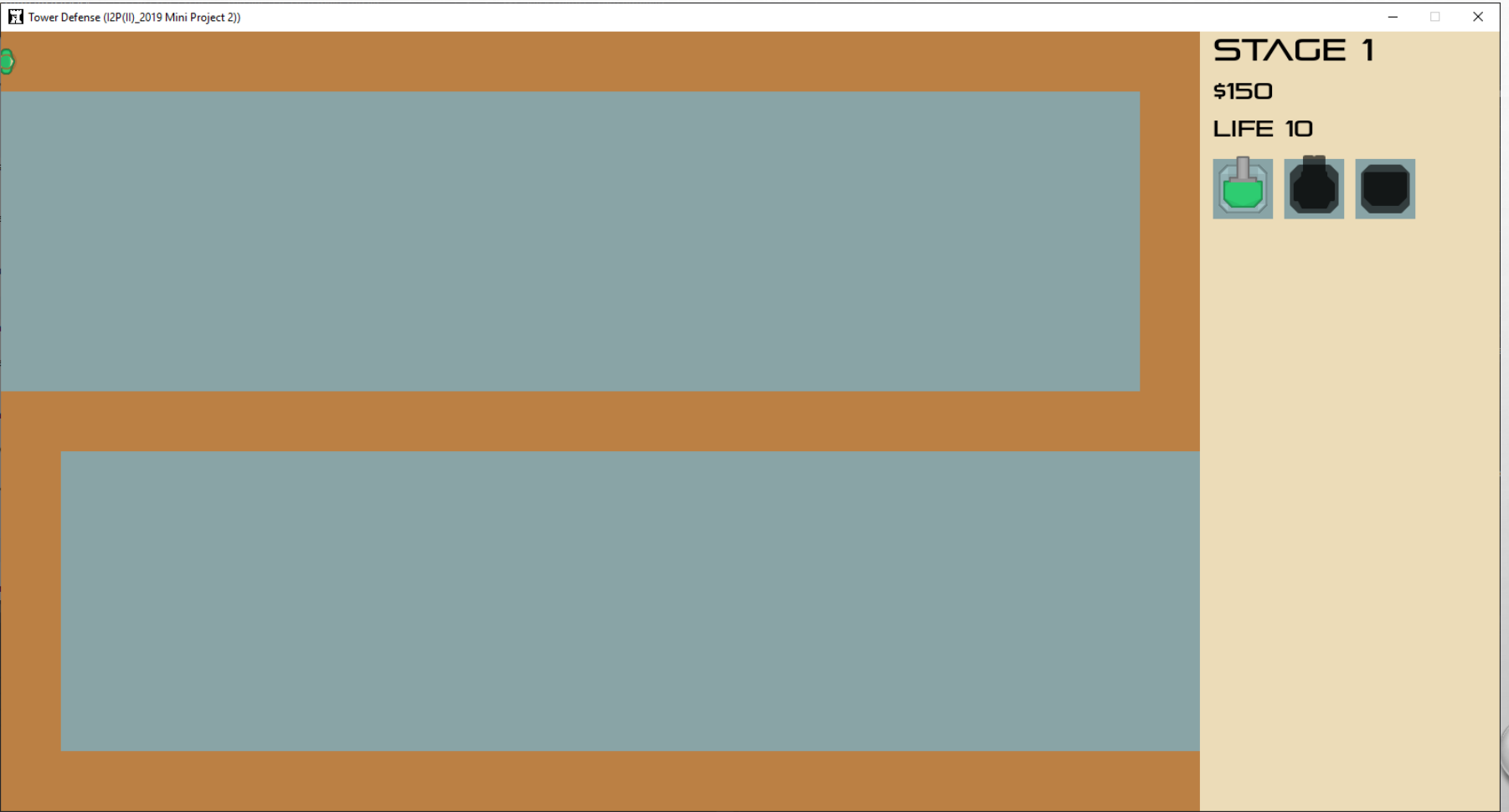
Game code reminder

Tower Defense

Reminder

- It would be easier to start from TODO 1 with just imitating the lose scene that mentioned in the code
- While coding BFS, a way you can traverse the map is simple using the parameter PlayScene::directions

Map file format



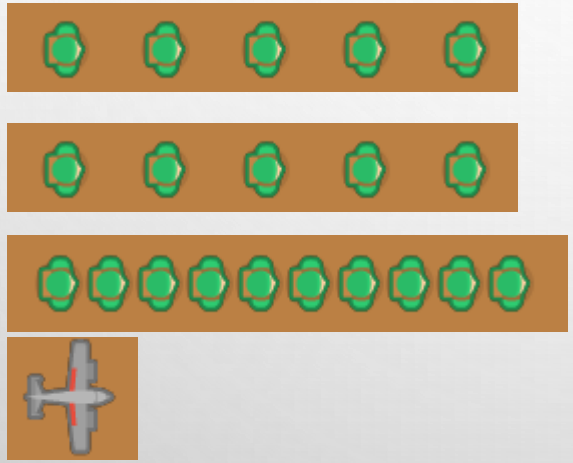

```
00000000000000000000
11111111111111111110
11111111111111111110
11111111111111111110
11111111111111111110
11111111111111111110
00000000000000000000
01111111111111111111
01111111111111111111
01111111111111111111
01111111111111111111
01111111111111111111
00000000000000000000
```

resources/map1.txt

Enemy file format

- EnemyType TimeDelayBetween Count

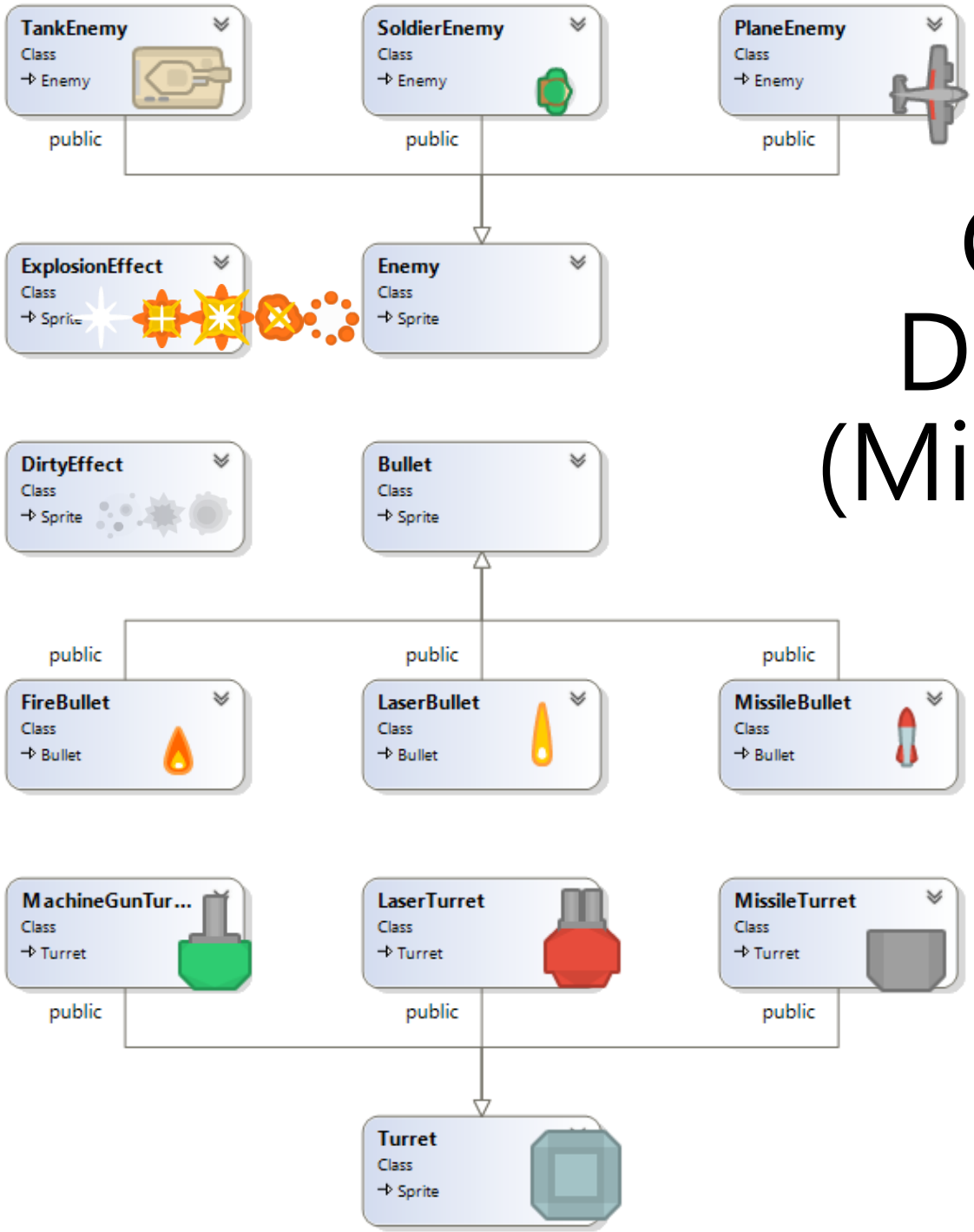
```
1 1 5
0 2 1
1 1 5
0 2 1
1 0.5 10
0 6 1
2 1 1
0 2 1
1 0.5 20
0 12 1
2 1 5
0 2 1
1 0.5 20
```

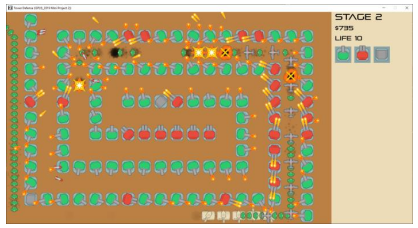


You should edit this file after adding new enemy

resources/enemy1.txt

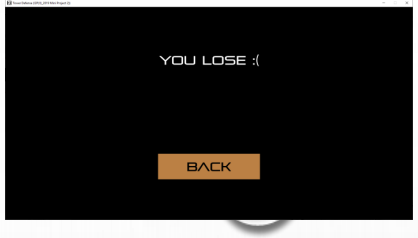
Game Diagram (Minimized)





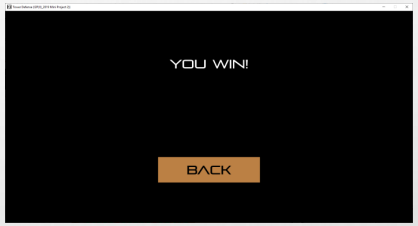
PlayScene
Sealed Class
IScene

LoseScene
Sealed Class
IScene



StageSelectScene
Sealed Class
IScene

WinScene
Sealed Class
IScene



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.

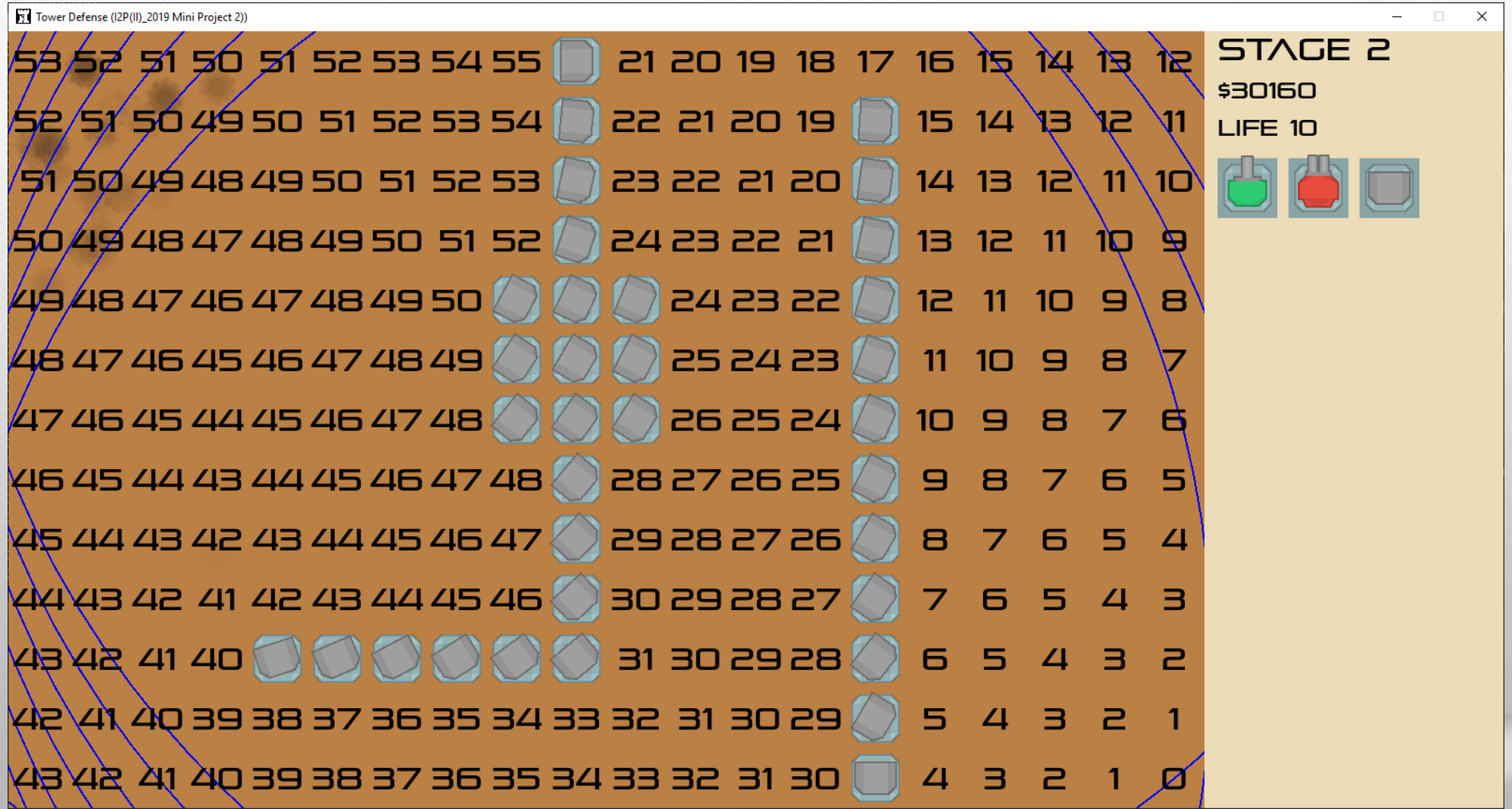
Grading Policy (2/5)

- 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)

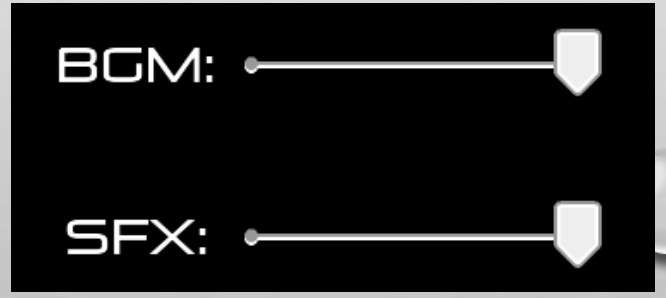
- Enemy path finding. (BFS) (1%)
  - For stage 4, the enemy requires path finding function to move towards our base.
  - Try to implement a simple BFS counting distances between any block and our base.
  - Press **TAB** can launch the debug mode to debug easier.
  - In the next page, you can see a example of calculating the distance from the endpoint(right bottom)

Grading Policy (3/5)



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: 6/11 (Tuesday)
- Time: 1320-1510
- Place: 資電館
- Grade: 5%

Week 11	4/30	Quiz 3: C++ basic	5/3	C++ polymorphism
Week 12	5/7	C++: Iterator	5/10	Midterm 2 review
Week 13	5/14	Midterm 2	5/17	Mini-project 2: window programming
Week 14	5/21	C++: template and STL	5/24	Problems using STL
Week 15	5/28	AI search using STL	5/31	Mini-project 3: AI games
Week 16	6/4	Quiz 4: STL and search problem	**6/7**	端午節放假
Week 17	6/11	Mini-project 2 demo	6/14	Review of final
Week 18	6/18	Final (由中午12考)	6/21	
	6/25	Mini-project 3 demo		

Thanks for your listening