# TECHNICAL OVERVIEW

# 

# 

# CODING METHODS

**File naming convention :** Create a file name by writing the first part of the word in capital letter. Also, writing the word which is abbreviation, in capital letters. (ex. SpriteBatch.cpp, FileIO.cpp)

**File location :** Files with similar functions are grouped and stored in same folder location. (ex. FileIO, ImageIO, KeyboardInput and MouseInput files of IO\_Manager folder)

**Code format :** C++, Visual Studio 2019

* **Variable naming convention :** Write word in small letters and put ‘\_’ between word and word. Also, write the word which is abbreviation, in capital letters. (ex. show\_another\_window)

* **Private variable and Define naming convention :** Write word in capital letters and put ‘\_’ between word and word. Also, write the word which is abbreviation, in capital letters. (ex. SCREEN\_WIDTH)

* **Function and Structure naming convention :** write the first part of the word in capital letter. Also, write the word which is abbreviation, in capital letters.

(ex. ChangeWindowType)

* **Different Coding Style :** Even if we tried to keep the above as much as possible, but there are still different coding styles for each member.

**Code documentation :** None

**Source control system :** Source tree

* Rule : Push the work with simple description of their work and CIB(Check In Buddy).

# DEBUGGING TOOLS

* Do you have an in-game debug console (or at least a simple output-only console)?

**YES. We use ImGui for debugging of our game.**

* Do you have a debug drawing system?

**NO. Our game is not platform or puzzle game genre. So, we don’t need drawing system for debugging.**

* Do you have a clean assertion system?

**No. Our debugging tool has not clean assertion system yet. But, we have a plan to implement it.**

* Do you have an in-game performance viewer?

**No. We don’t have in-game performance viewer.**

* Do you have a way to watch variables in-game?

**YES. Even though we don’t need, we can see the RGB of the background color and number of times button has been pressed.**

* Can you shut off your debug controls?

**YES. we can close the debugging tool by clicking the ‘X’ button of window of debugging tool.**

# GRAPHICS OVERVIEW

**Our graphics engine is SDL based 2D basic game engine.**

Make sure these questions are answered:

* What techniques will be used to implement the graphics design?

**Window** : SDL based window setting

**Mouse & Keyboard** : SDL based key input

**Shader :** GLEW based to make points and colors.

**Texture :** IMAGE LOAD class to implement the basic TEXTURE class and used the external library pico PNG.

**Sprite** : GLEW based to creating positions and uv values of points, and color and texture depending on the number of points.

**Mesh :**

* What graphics API will you use?

**SDL2, GLEW**

* Are you using fixed function or shaders?

**In our graphic, we make a shader and use it as a shader every time we use different sprites.**

* How are you loading assets such as sprites, models, textures, and animations?

**Our pipelines implement the texture in IMAGE\_LOAD class. And we use an external library, pico PNG. To add art(sprite, texture, model) to the program, you must re-compile it by adding one function. This is an automated program. Animation is also made from simple sprite based uv calculations.**

**Sprite** : use the picoPNG and sdl\_image

**Models** : use the Sprite class

**Texture** : use the picoPNG and sdl\_image

**Animations** : use the Sprite class

# PHYSICS OVERVIEW

Show/describe the structure of the physics engine.

**Our engine use very simple physics**

Make sure these questions are answered:

* What techniques will be used to implement the physics design?

**We don't use specific techniques. Because we only use mouse collision in our game.**

* What type of integration will you use (Euler, Improved Euler, Verlet, Runge-Kutta)?

**We don't use integration. Because we only use mouse collision in our game.**

* What type of space partitioning will you use?

**We don't use space partitioning. Because we only use mouse collision in our game.**

* What type of collision detection will you use?

**We only use Mouse Collision in our game.**

# BEHAVIORS OVERVIEW

Show/describe the structure of the AI/behaviors in your game.

**AI of our game is very simple. It chooses it’s behavior at random. Also, our behaviors are made up of just texture change or animation effects.**

Make sure these questions are answered:

* What types of algorithms will be used to implement the behavior design?

**We use random algorithm for enemy’s behavior. This algorithm generates random 3 behaviors of enemy.**

* Pattern movement? Pathfinding? State machines? Flocking? Influence maps? Genetic algorithms?

**We don’t use above techniques. We just use random method.**

# MULTIPLAYER OVERVIEW

Show/describe the structure of how multiplayer will work in your game.

**Our game support only single play.**

Make sure these questions are answered:

* What techniques will be used to implement multiplayer?

**We don't use specific techniques. Because our game support only single play.**

* If your game is networked, how will you host games and detect players?

**We don't use network. Because our game support only single play.**

* What networking protocols will you use?

**We don't use networking protocols. Because our game support only single play.**

* Will you use encryption, data compression, host migration, etc.?

**We don't use above things. Because our game support only single play**

# EDITOR OVERVIEW

Show/describe the structure of how your editor works.

**We don’t support editor.**

* How do you create/delete/copy objects?

**We don’t support editor.**

* How do you move/rotate objects?

**We don’t support editor.**

* How do you save/load levels?

**We don’t support editor.**

* How do you update archetypes?

**We don’t support editor.**

# ADDITIONAL TOPICS

* **Art pipeline**

Our art pipeline has more than two error checks, and does PNG load. Our pipelines implement the texture in IMAGE\_LOAD class. And we use an external library, pico PNG. To add art to the program, you must re-compile it by adding one function. This is an automated program. However, it is really easy to delete and add art in that it works as a function. And we haven't made drag and drop yet . VFX is too.

* **Audio pipeline**

Our team's audio pipeline is based on SDL\_MIXER and supports basic audio functions such as sound effects, background music, stop and start. Our audio does not use FMOD or FMOD Studio. It should be compiled again to add or modify audio. Adding and deleting audio is also easy with the function INIT. Our audio is out of real-time control, but there are plans to make it. Audio drag and drop is also not supported. The audio report processing is neat. Our audio reads the path and file contents of the audio file in SDL\_MIXER and determines when it will be processed.

* **Testing tools**

Our team use ImGui for debugging tool of our game. we don't have debug drawing capability and error messages. Also, we don't have autoplay feature, gameplay recording and playback features. Moreover, we don't have data-tracking system. However, we have cheat code which player can win the game immediately.

**CHEAT CODE :** If you press ‘c’, bear’s attack power increase 5 to 50.