FINAL PROJECT RULES

Introduction to Programming 2020

Salamander (1986)

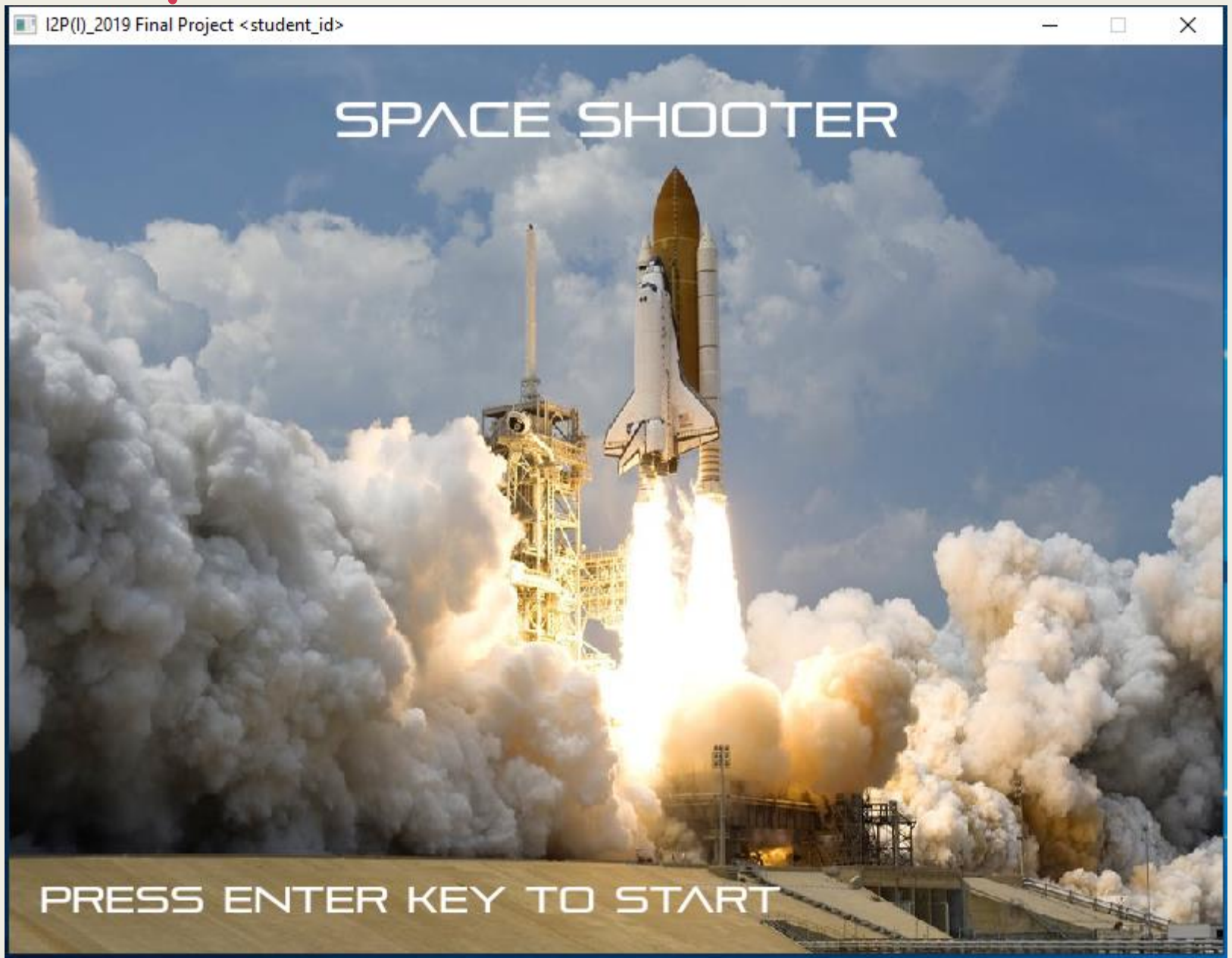
Raystorm (1996)

東方永夜抄 ~ Imperishable Night. (2004)

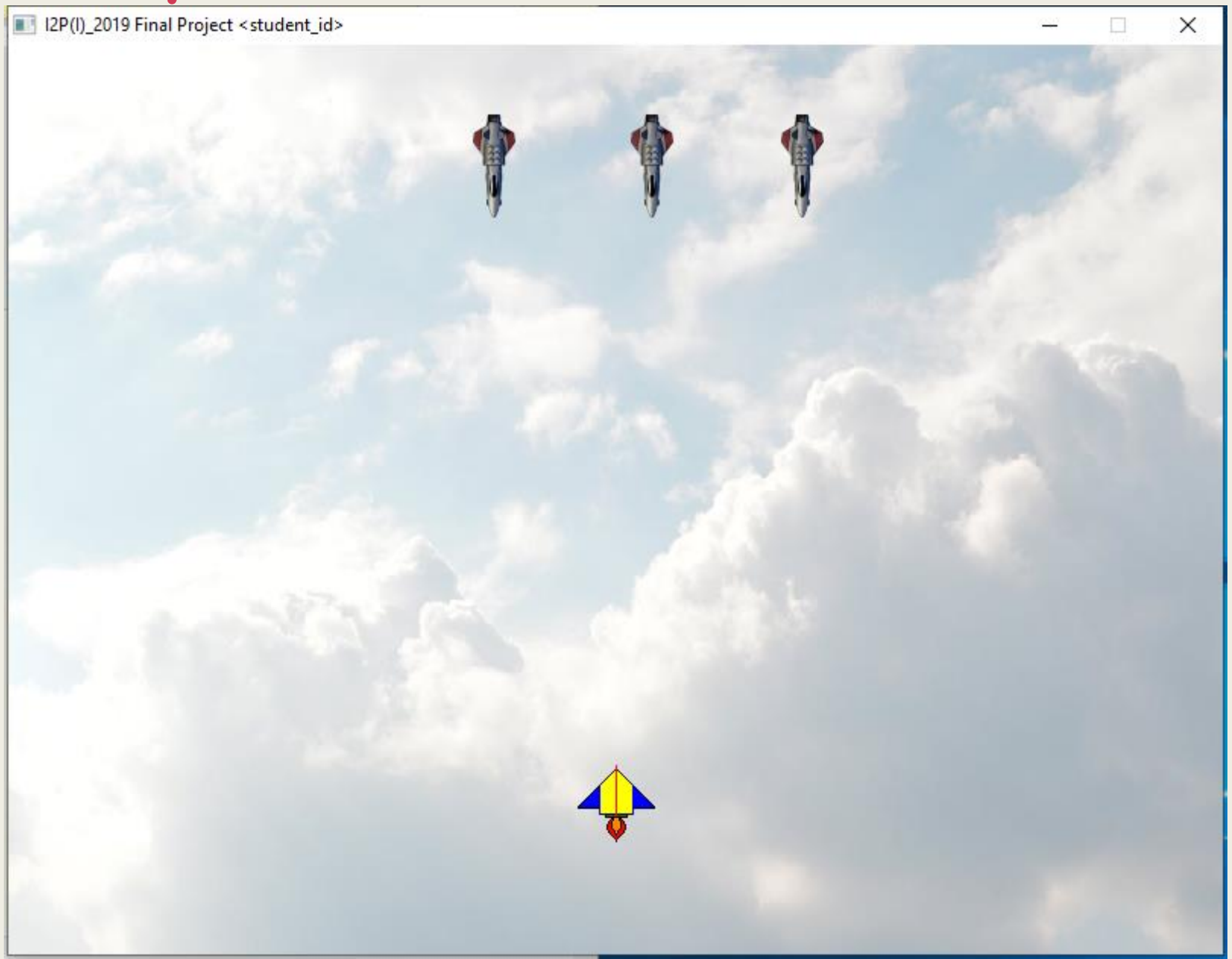
Rules

- One person per group
- Worth 15% of your total grade.
- Use Allegro 5 library finish a 2D plane shooter game independently.
- Must use the template we provided.
- 2021.01.18 (Mon) Demo
  - Use your own computer to demo
  - More details will be announced one week before demo.
- Can only use C, no C++ or python.

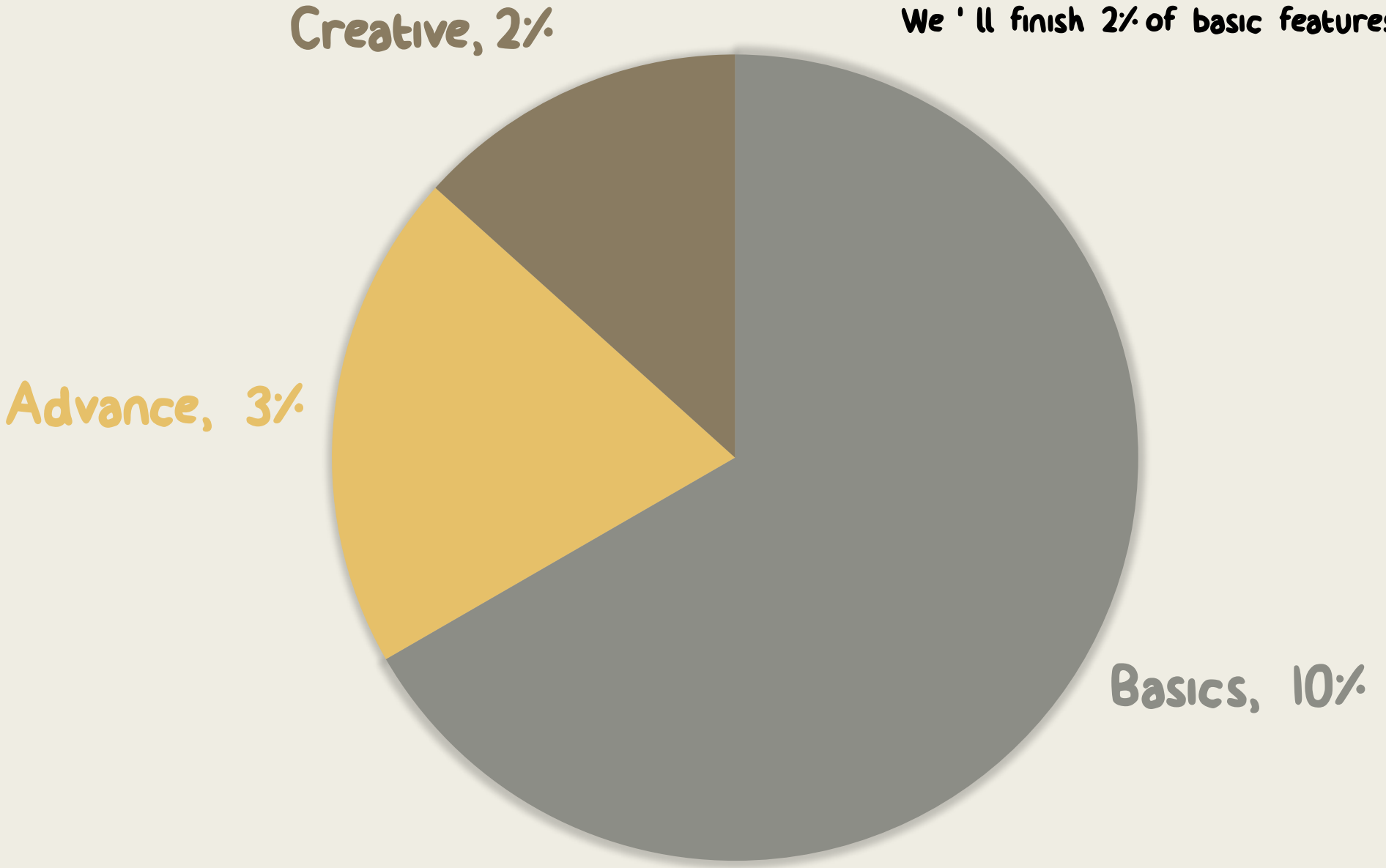
Given template



Given template



We ' ll finish 2% of basic features today



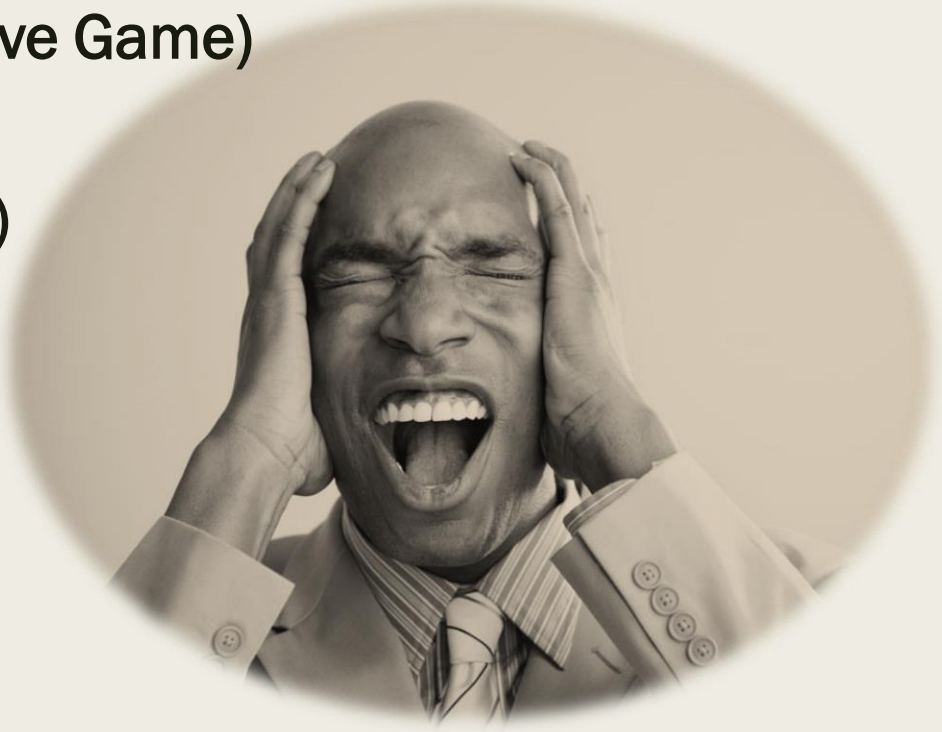
Basics (10%)

- The completeness of your game.
- Scoring in game.
- Plane (cannot exceed the display boundaries), Enemies、Bullets can move correctly.
- Bullets can deal damage.
- Health Bar (HP) / Remaining lives (lives)
- Use mouse and keyboard
- The 4th scene (we already have Menu, Start, Settings)
  - e.g., Win, Game over, Restart, End, …
- 2% for today (the features marked in red.)

Advance (3%)

- Opening Animation + Character Animation
- Permanent Scoring (High Score / Save Game)
- 2.5D Scene
- 2P Mode (Cooperate shoot enemies)
- Character Select
- Sound Effects + Background Music (different music in different scene)
- Boss
- Ultimate (Ult) / Special Attack

Choose 3 to implement.



Creative (2%)

- Character appearance
- Magnificence attack
- Cool animations
- Richness of your game
- Good Performance (no lags)
- .....

Template

- Single file template
  - *Final Template (basic).zip*
  - Easier to get used to, but it would be messy when you implement too many features.

- Multiple file template
  - *Final Template (advanced).zip*
  - Harder to get used to, but functions and scenes are separated to different files.

Template (if you use Allegro 5.0)

- Change

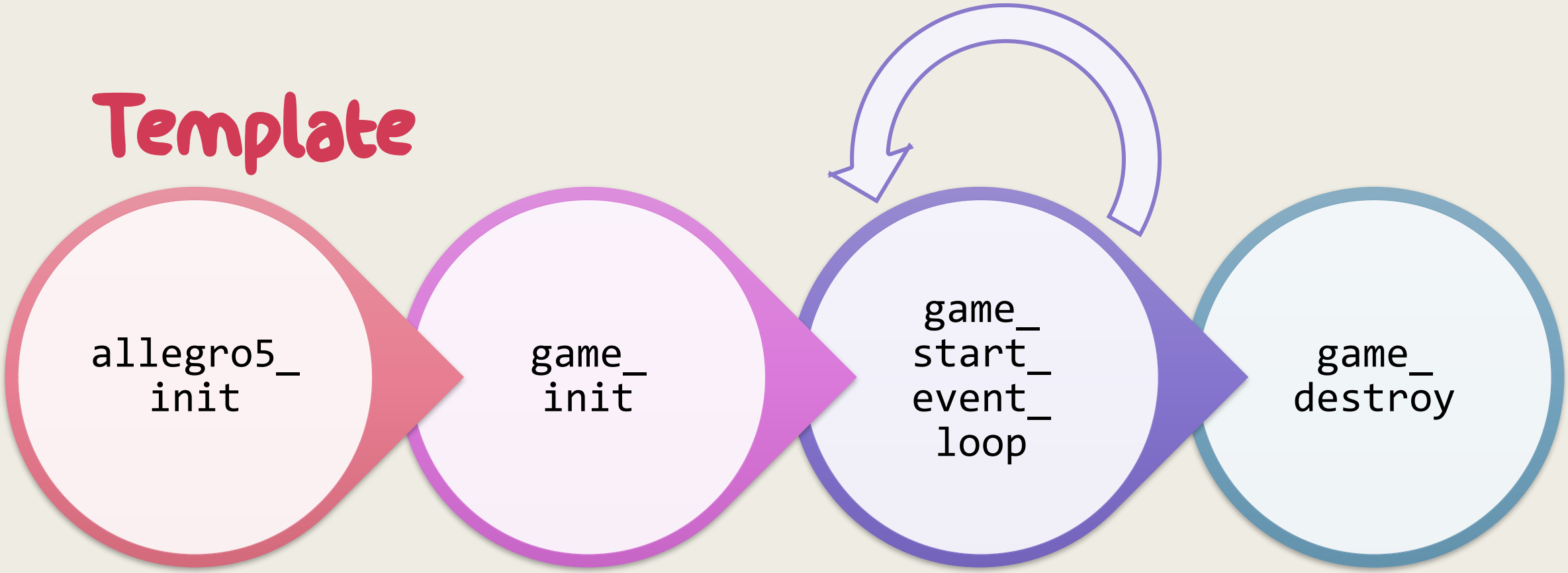
```
if (!al_init_font_addon())
    game_abort("failed to initialize font add-on");
```

to

```
al_init_font_addon();
```

- (You only need to fix this if you followed the tutorial that uses **`allegro-5.0.10-monolith-mt.dll`**)

Template



Init lib routines
- init/install
- create display, event queue, timer
- register events
- start timer

Init variables
- load resources
- change scene
  - to main scene

Process events
- close window
- timer
  - update
  - draw
- keyboard events
- mouse events

Free variables
- free resources
- change scene to main scene

Template (states)

```
// The active scene id.
int active_scene;
// Keyboard state, whether the key is down or not.
bool key_state[ALLEGRO_KEY_MAX];
// Mouse state, whether the key is down or not.
// 1 is for left, 2 is for right, 3 is for middle.
bool *mouse_state;
// Mouse position.
int mouse_x, mouse_y;
```

Template (structs)

```
typedef struct {
    // The center coordinate of the image.
    float x, y;
    // The width and height of the object.
    float w, h;
    // The velocity in x, y axes.
    float vx, vy;
    // Should we draw this object on the screen.
    bool hidden;
    // The pointer to the object's image.
    ALLEGRO_BITMAP* img;
} MovableObject;
```

Template (routines)

```
// Initialize allegro5 library
void allegro5_init(void);
// Initialize variables and resources.
void game_init(void);
// Process events inside the event queue using an infinity loop.
void game_start_event_loop(void);
// Release resources.
void game_destroy(void);
// Function to change from one scene to another.
void game_change_scene(int next_scene);
```

Template (events/callbacks)

```
// This is called when the game should update its logic.
void game_update(void);
// This is called when the game should draw itself.
void game_draw(void);
void on_key_down(int keycode);
void on_mouse_down(int btn, int x, int y);
```

Template (utilities / callbacks)

```
void draw_movable_object(MovableObject obj);
// Load resized bitmap and check if failed.
ALLEGRO_BITMAP *load_bitmap_resized(const char *filename, int w, int h);
// Display error message and exit the program, used like 'printf'.
// Write formatted output to stdout and file from the format string.
// If the program crashes unexpectedly, you can inspect "log.txt" for
// further information.
void game_abort(const char* format, ...);
// Log events for later debugging, used like 'printf'.
// Write formatted output to stdout and file from the format string.
// You can inspect "log.txt" for logs in the last run.
void game_log(const char* format, ...);
```

Template (Advanced version)

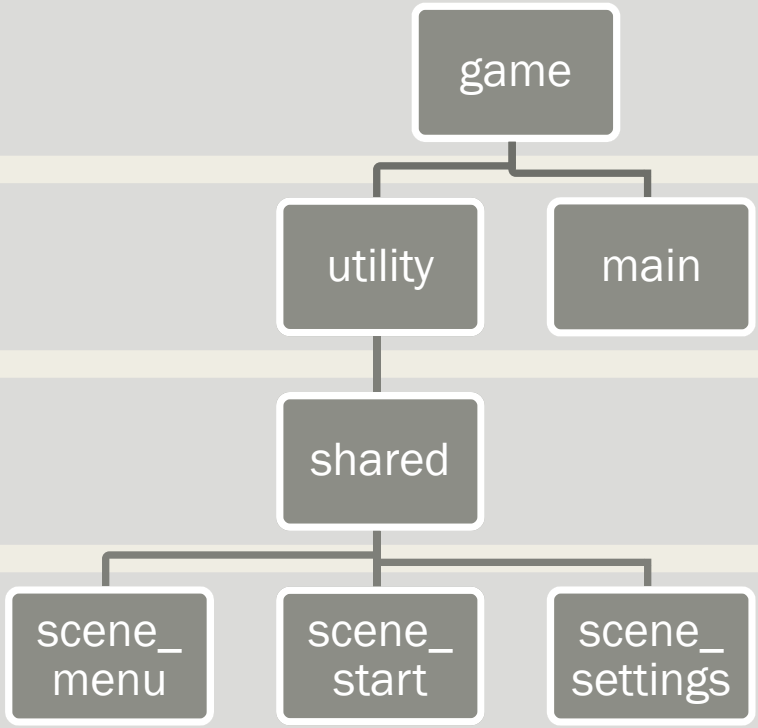
- If you want to use multiple files.

Allegro5 routines &
Scene control

Utility functions &
Entry point

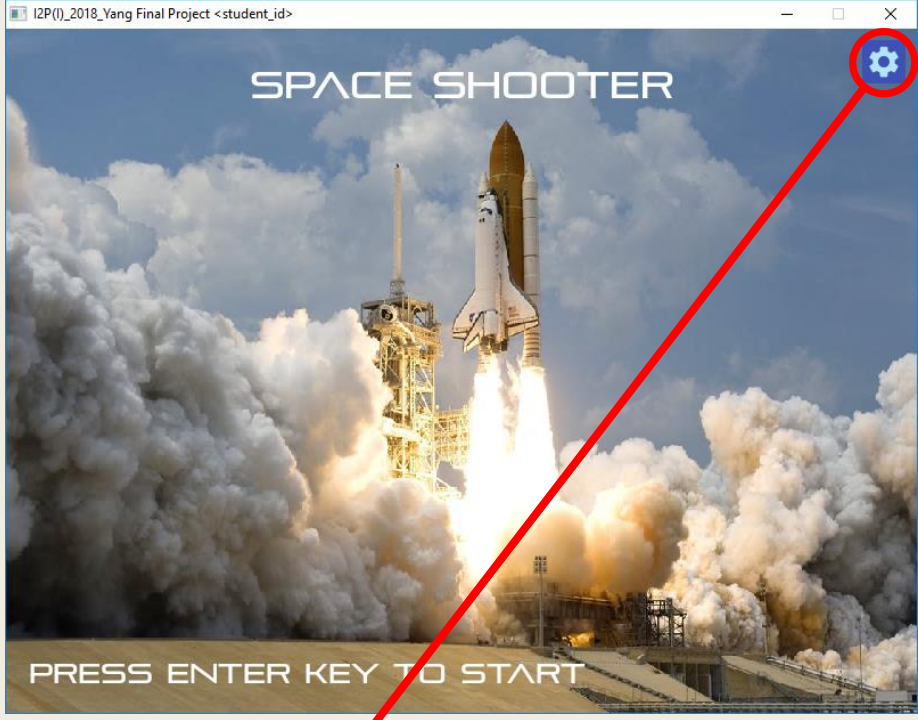
Shared resources

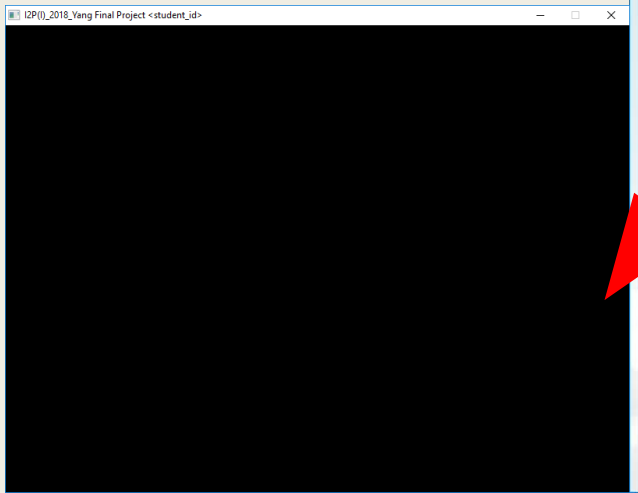
Scenes

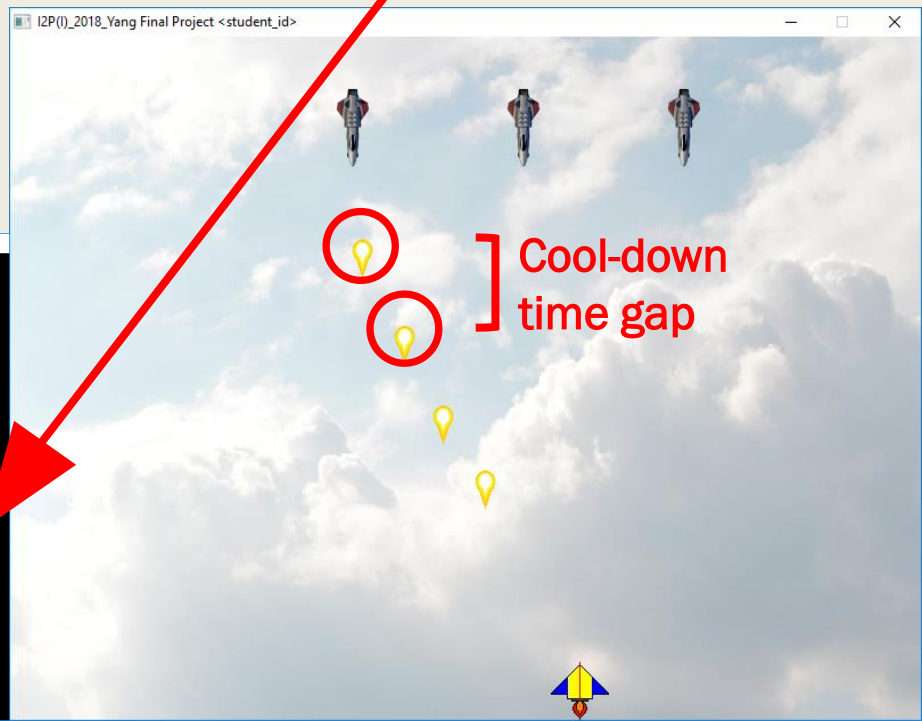


Today 's Goal

- Setup boundaries for your airplane / rocket
- Shoot multiple bullets with cool-down
- Implement a new scene
  - *Create the settings scene. (can be entirely black with no functions)*
  - *A button in main scene. (w/ mouse in/out animation)*







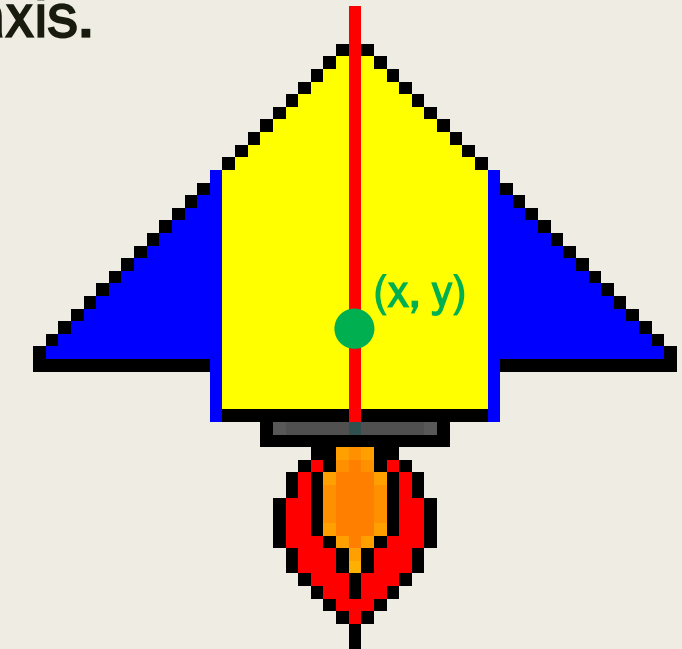
Today 's Goal (Example)

- For today’s goal, you only need to uncomment the codes and replace the “???” with the correct code.

```
// [HACKATHON 1-1]
// TODO: Limit the plane's collision box inside the frame.
// (x, y axes can be separated.)
// Uncomment and fill in the code below.
//if (??? < 0)
// plane.x = ???;
//else if (??? > SCREEN_W)
// plane.x = ???;
//if (??? < 0)
// plane.y = ???;
//else if (??? > SCREEN_H)
// plane.y = ???;
```

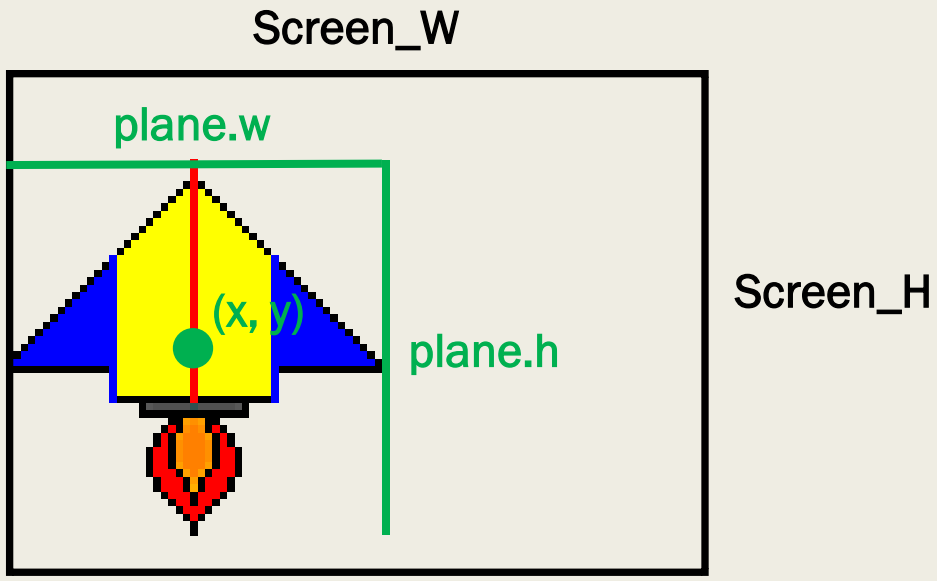
Today 's Goal (1)

- Setup boundaries for your airplane / rocket
- TODOs: [HACKATHON] 1-1 ~ 1-1
- Separate the x and y axes. Use the same calculation to detect each axis.



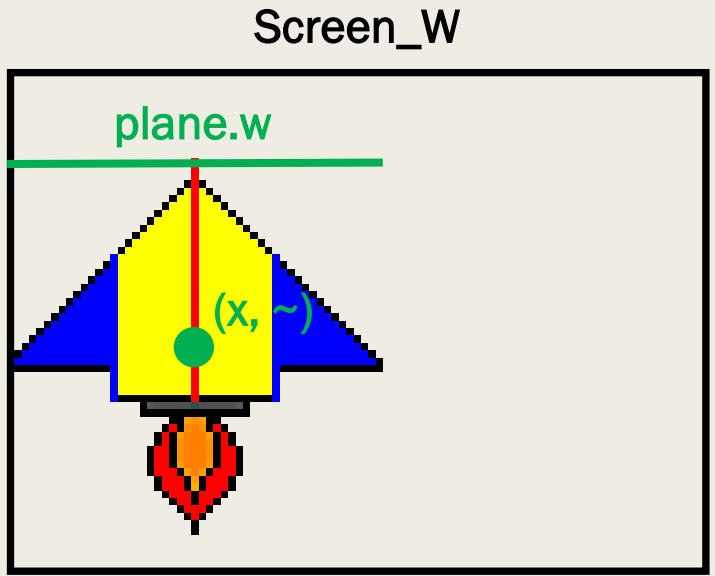
Today 's Goal (1)

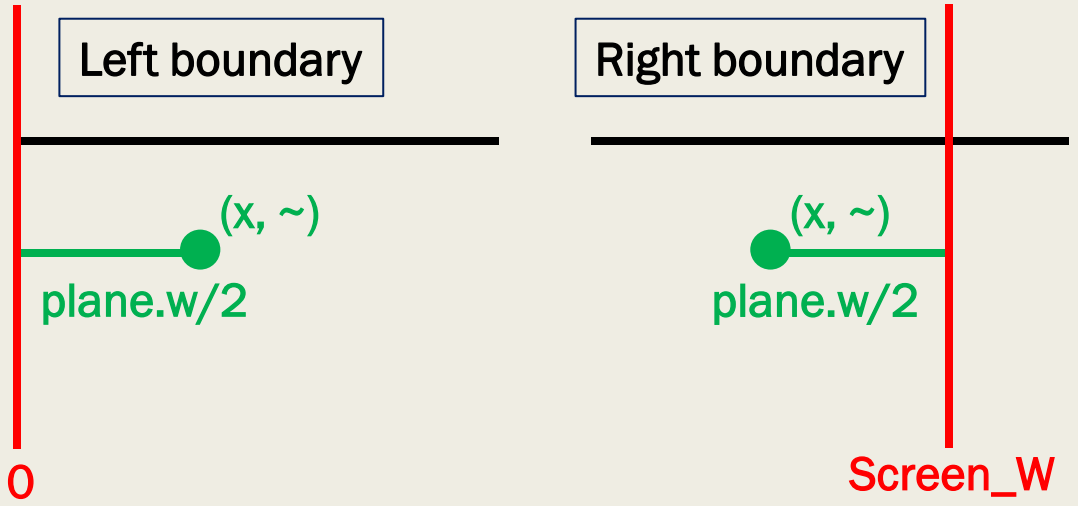
- Setup boundaries for your airplane / rocket
- TODOs: [HACKATHON] 1-1 ~ 1-1
- Separate the x and y axes. Use the same calculation to detect each axis.



Today 's Goal (1)

- Setup boundaries for your airplane / rocket
- Separate the x and y axes. Use the same calculation to detect each axis.
- Left bound and right bound for x-axis.



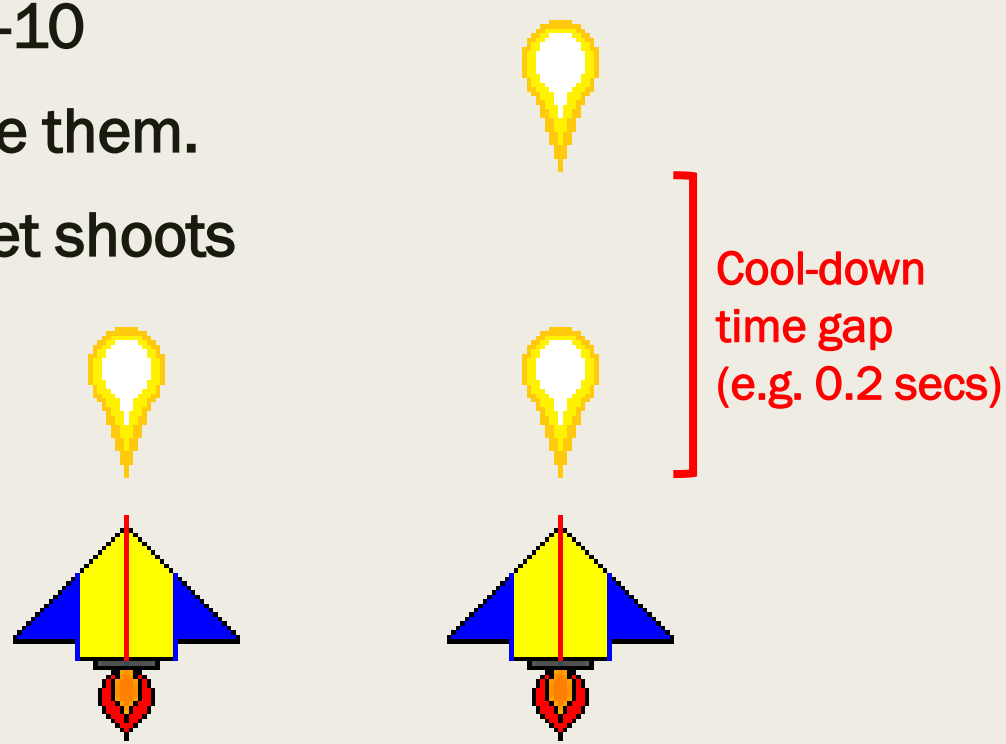


Today 's Goal (II)

- Shoot multiple bullets with cool-down
- TODOs: [HACKATHON] 2-1 ~ 2-10
- Create a bullet array and reuse them.
- Control the time between bullet shoots

Bullets (h for hidden)

		h		h	h	h	h	h	h



Today 's Goal (III)

- Implement a new scene
  - *Create the settings scene. (can be entirely black with no functions)*
  - *A button in main scene. (with mouse in/out animation)*
- TODOs: [HACKATHON] 3-1 ~ 3-10

In game_change_scene, game_update, game_draw, on_key_down, ...

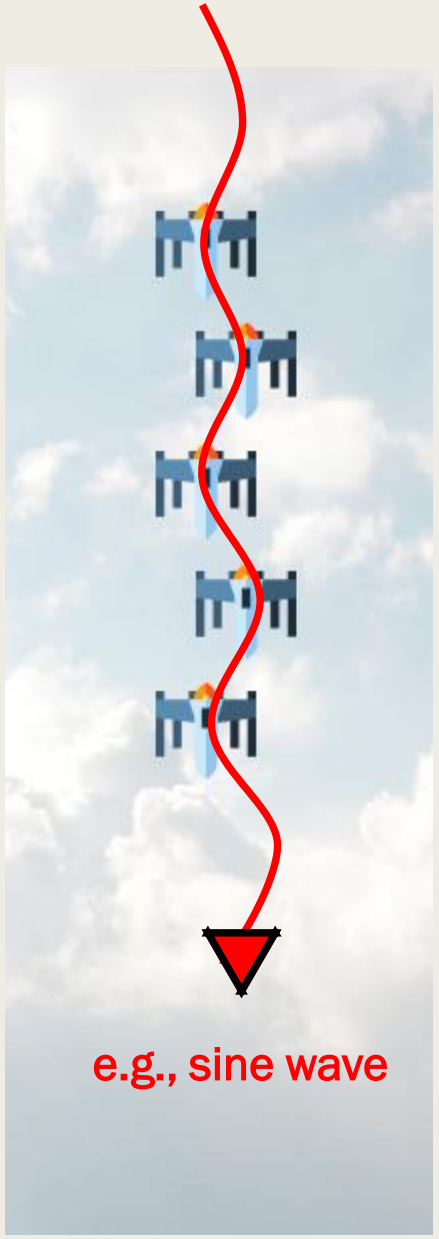
```
if (active_scene == SCENE_MENU) {
    //…
} else if (active_scene == SCENE_START) {
    //…
} else if (active_scene == SCENE_SETTINGS) {
    //…
}
```

Today 's Goal (IV) (Bonus!)

- Implement enemy AI and collision mechanism
  - *Randomly spawn enemy groups.*
  - *Enemy can move along a nonlinear trajectory in groups.*
  - *Reuse enemies when exceeding the screen boundaries.*
  - *Bullets can kill enemies. (multi-shot kill, no one-shot kill)*
  - *If enemies hit our plane, exit. (go back to menu scene)*

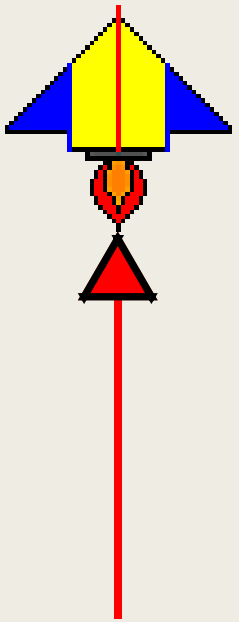
- TODOs: None (Implement from scratch!)

Hint: Derive x directly by linear increasing y
or Follow trajectory with constant speed by
deriving the velocity by differentiation



Today 's Goal (V) (Bonus!)

- Implement new enemy with rotating homing bullets
  - *Randomly spawn single enemies.*
  - *Enemies shoot rotating homing bullets continuously. (the acceleration should not guarantee 100% hit)*
  - *If bullets hit our plane, exit. (go back to menu scene)*
  - *If our bullets collide with the enemies' bullets, both bullets should disappear. (be released & reused)*



- TODOs: None (Implement from scratch!)

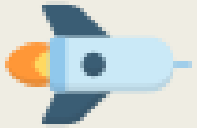
Today 's Goal (V) (Bonus!)

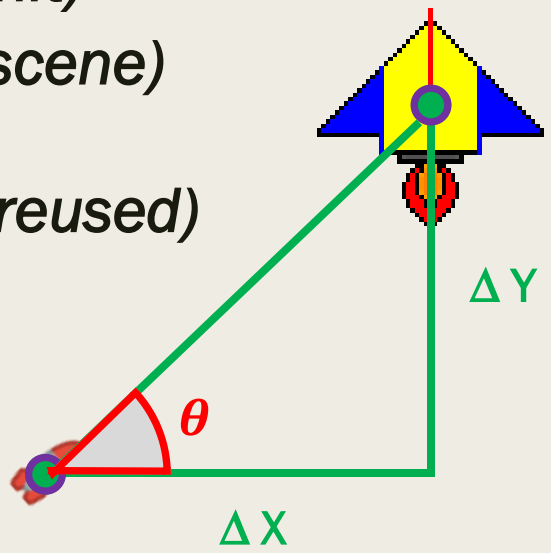
1. Rotate θ
2. Target velocity is $(\Delta x, \Delta y)$ by simple interpolation or by deriving the acceleration.

$$\theta = \tan^{-1}\frac{\Delta y}{\Delta x}$$

Hint: `theta=atan2(…)`

- Implement new enemy with rotating homing bullets
  - *Randomly spawn single enemies.*
  - *Enemies shoot rotating homing bullets continuously. (the acceleration should not guarantee 100% hit)*
  - *If bullets hit our plane, exit. (go back to menu scene)*
  - *If our bullets collide with the enemies' bullets, both bullets should disappear. (be released & reused)*
- TODOs: None (Implement from scratch!)



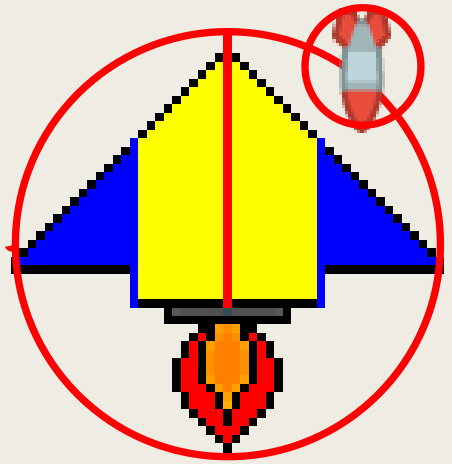


Today 's Goal (VI) (Bonus!)

- Implement advanced collision detection mechanism
  - *Perform an overlapping test with bounding area (Circle/Box/Convex Hull)*
  - *If the test succeeds, perform pixelwise collision detection. (Need to consider image rotation)*

- TODOs: None (Implement from scratch!)

Hint: Read image pixels to determine the precise bounding area



The above objects do not collide

Today 's Goal (Total 2.5% Bonus!)

- Aside from filling the blanks, make sure you understand the entire game flow and how each code section works.
- Find a TA and demo the first 3 goals (Task I to III) to get 0.5% bonus score.
- The TA will ask you to explain how the first 3 goals (Task I to III) are implemented, you'll get 0.5% bonus score for describing how the code works.
- (Bonus!) Find a TA and demo the 3 bonus goals (Task IV to VI) to get a total of 1.5% (0.5% each) bonus score. The TA will ask you to explain your code.

Resources

- Kenny's Assets (Free)
  https://www.kenney.nl/assets?q=2d
  All assets are released in CC0 license!

- OpenGameArt (Free)
  https://opengameart.org/
  The licenses may vary

About Plagiarism

- We encourage your discussion during Hackathon and when implementing your final project.
- Do not copy others' code directly.
- We will perform comparison on your code for checking plagiarism. (including previous years' final project)
- If your code has high similarity with previous years' final project and you have no reasonable excuses, you may fail the class depending on the severity.

DON'T CHEAT

LET'S CODE

Have a nice day~