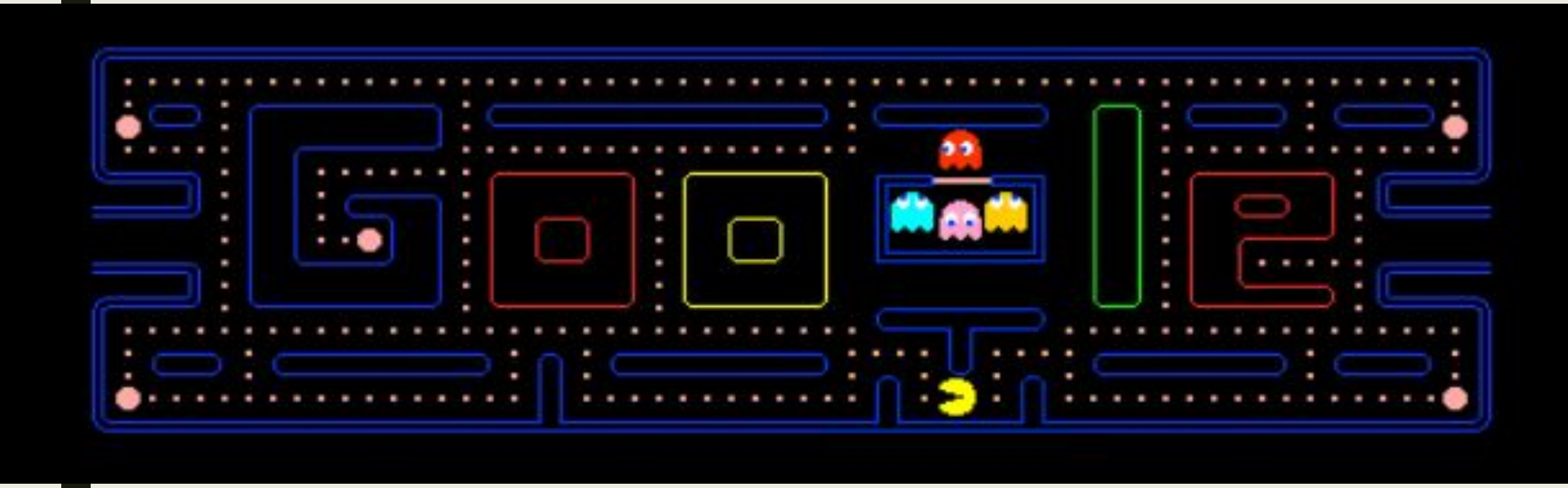
FINAL PROJECT RULES

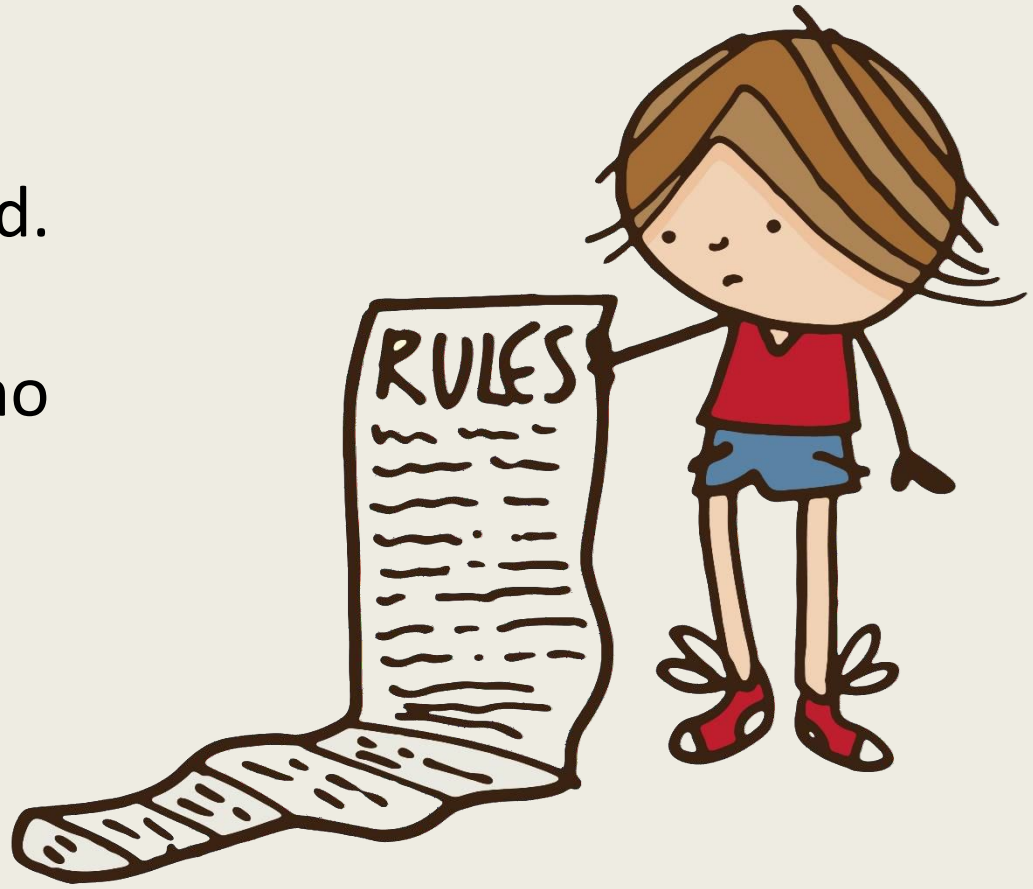
Introduction to Programming 2021


HI-SCORE
10000
440

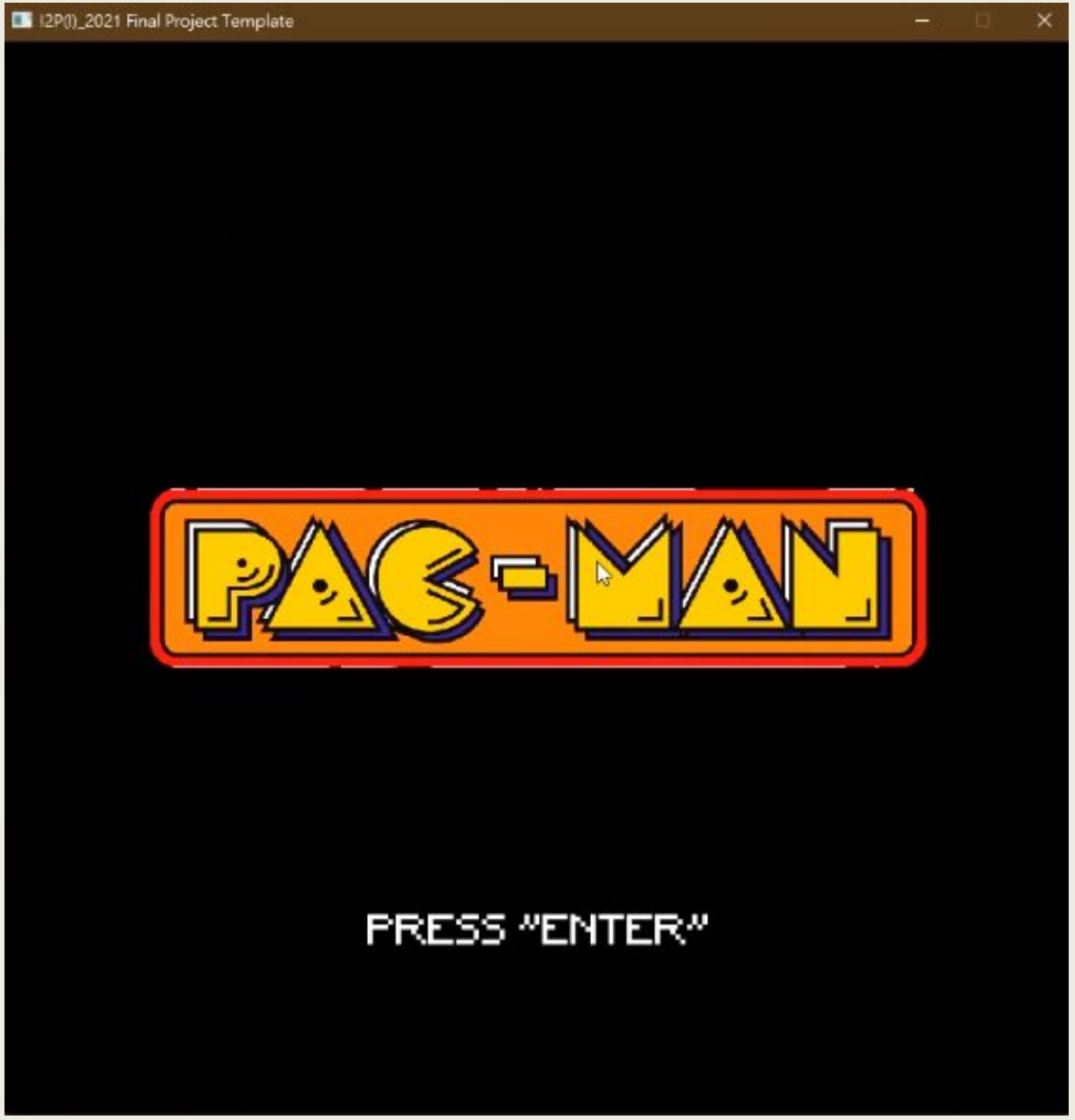


Rules

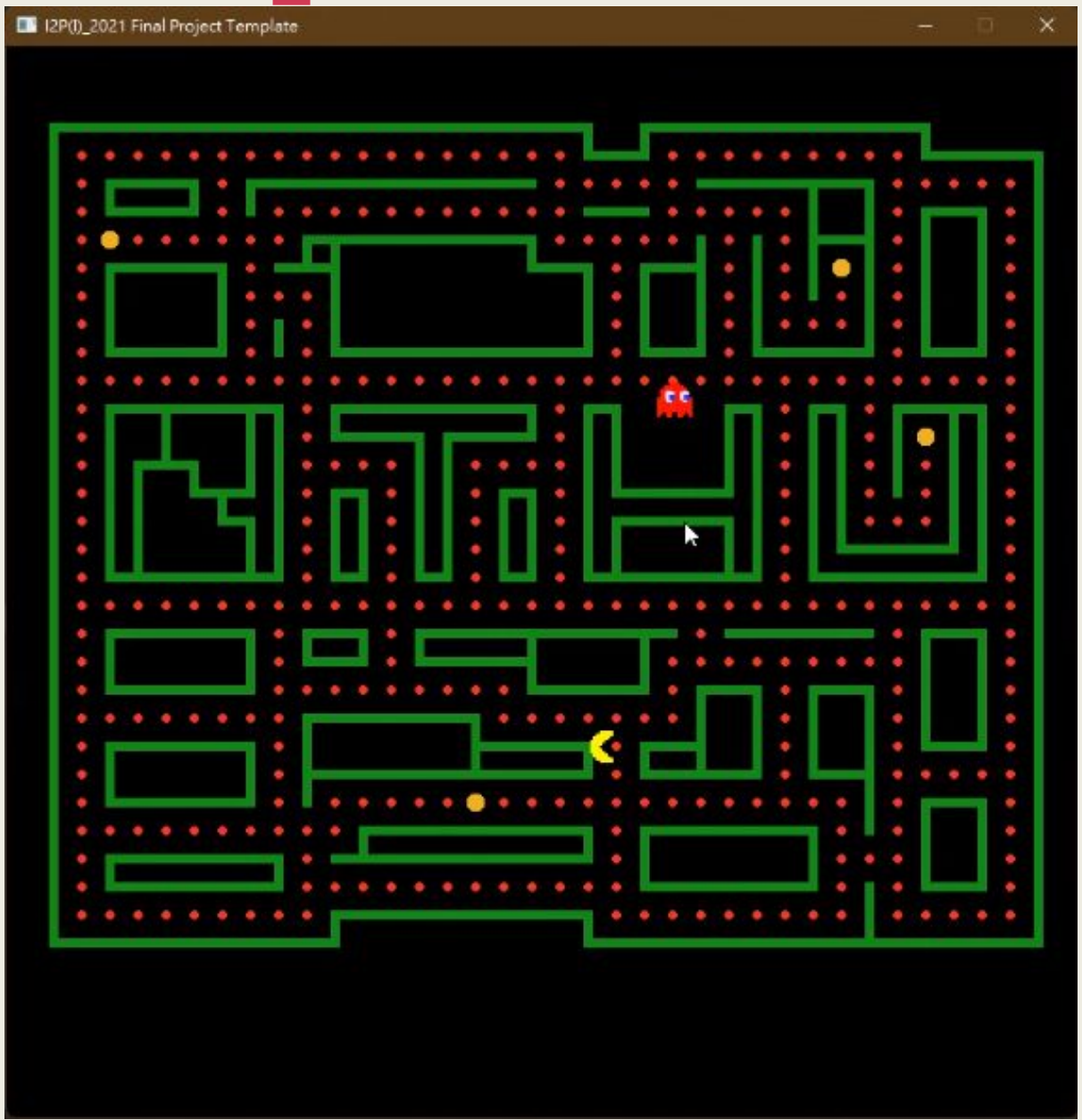
- One person per group
- Worth 20% of your total grade.
- Must use the template we provided.
- 2022年1月18、19日Demo
  - Use your own computer to demo
  - More details will be announced one week before demo.
- Can only use C, and boolean provided by allegro
  - No C++ or Python



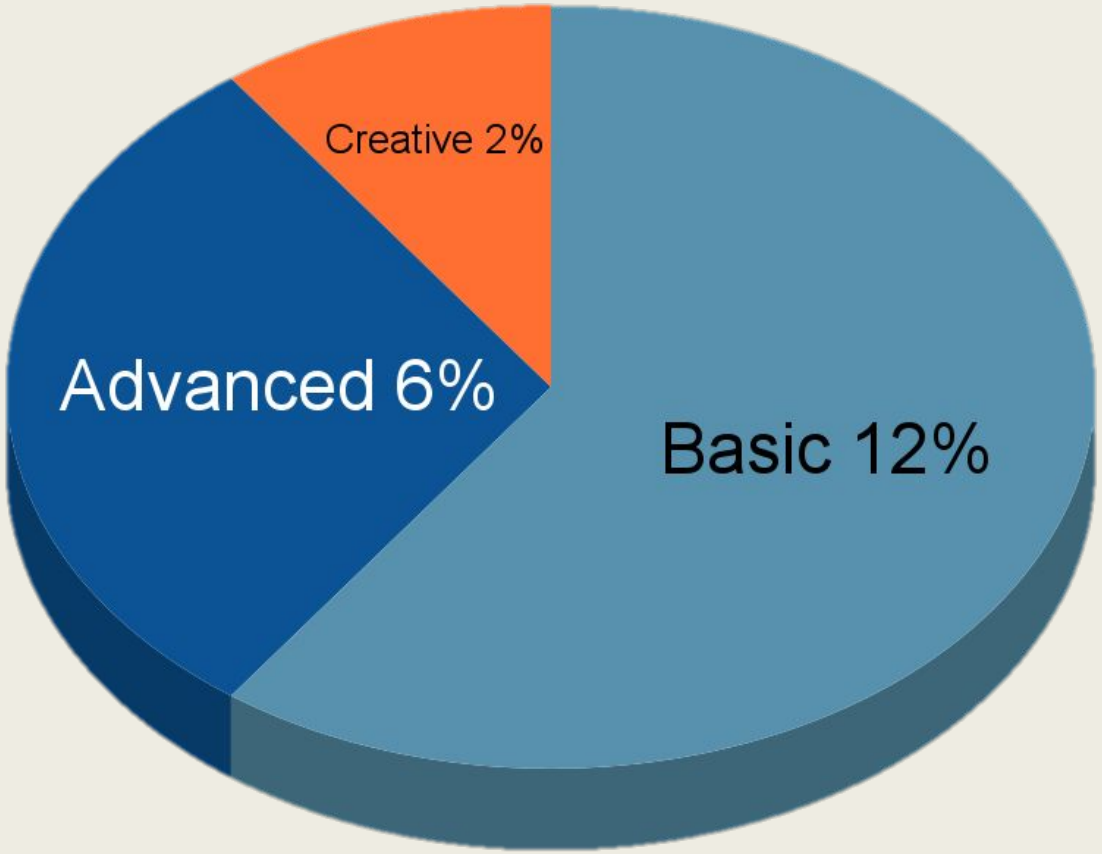
Given template



Given template



We’ ll finish 3% of basic features today



Basics (12%)

- **Divided into 5 parts.**
- **Game Completeness Part 3%**
- **Scene Part 2%**
- **Control Part 2%**
- **Memory Management 2%**
- **HACKATHON 3%**

Basics (12%)) Game Completeness Part(3%)

- Pacman [HACKATHON-1] ‘s movement (Can’t pass through wall or run into graphical error)
- Eat Beans [HACKATHON-1]
- Pacman should die if the pacman is touched by the ghosts
- Game should end normally after all the beans are eaten (Or start next round)
- Ghost
  - Ghost’s movement[HACKATHON-2] (Can’t pass through wall or run into graphical error)
  - Ghost go out of cage [HACKATHON-2]
- Read map from .txt files to generate map
- Score points when beans are eaten by the Pacman.
- Random movement Ghost Spec:
  - Should not repeat the same walking path. (No hard code)

Basics (12%) Scene Part(2%)

- The original three scenes : Menu, Game, Setting [HACKATHON-3]
  - Successfully switching between Scenes is required.
  - Should go back to menu or the next scene after the end of game scene
    - Program closes unexpectedly is unacceptable.
      - The only conditions of closing the program is when“close window" or self designed EXIT UI is clicked.
      - Add a 4^{th} scene (we already have Menu, Start, Settings)
        - e.g Win, Game Over, Restart, End, etc.

Basics (12%)Control Part(2%)& memory management(2%)

- Use mouse (ex. click and enter different scenes [HACKATHON-3] and keyboard (pacman controls) events[HACKATHON-1]
  - Volume adjustment in the Settings Scene
- Memory management
  - Memory Usage is bounded
  - Just make sure everything you allocated are deleted when the program finishes.
  - (We will use profiler to test this.)

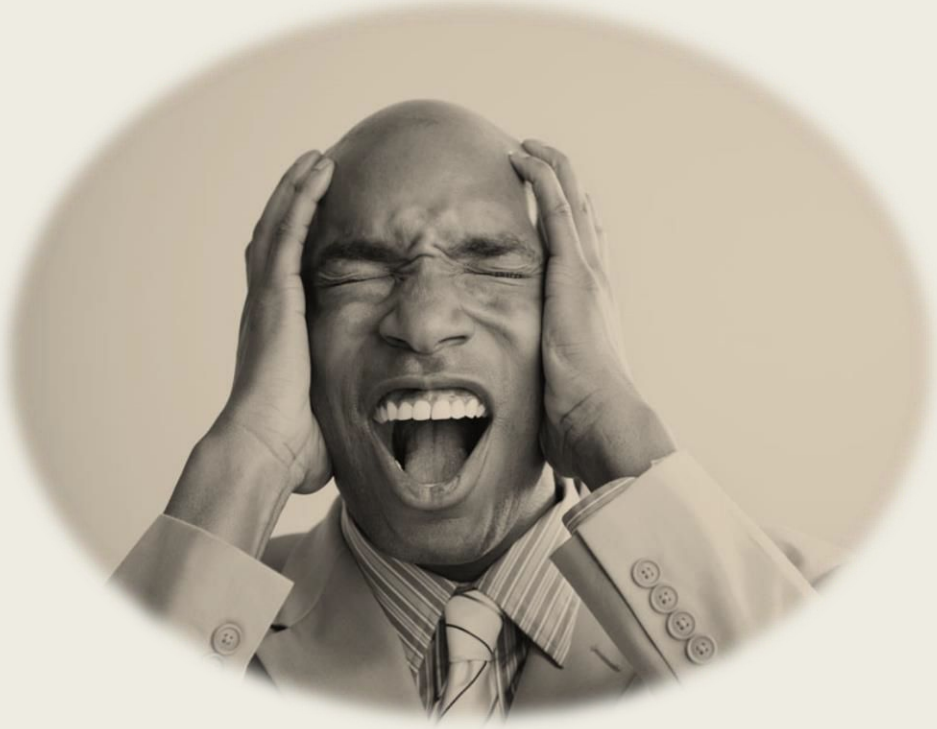
Basics (12%) HACKATHON(3%)

- (Those marked in red in previous slides)
- HACKATHON 1
  - Pacman movement and eating beans
- HACKATHON 2
  - Ghost leaves the cage with random movement
    - Current version is hard coded.
- HACKATHON 3
  - Enter Setting scene by using the mouse

Advance (6%)

- Power Bean (2%)
- Design tracking rule of other Ghosts(1%)
- Character Animation (ex. Pacman's mouth, Ghost movement) (1%)
- Art (1%)
- Gameplay(1%)
- Function、Interface Part(1%)

(Note: Sum up to at most 6%)



Advance (6%)Power Bean (2%)

- Implement Classic Pacman power bean functionality
  - After eating the power bean, pacman can eat the ghosts for a certain amount of time. Ghost should become blue (sprites images are provided) and move slowly in this period.
  - When the power effect is running out, ghosts should twinkle blue and white.
  - Ghosts run away from pacman in this period.
  - Ghost should go back to the cage if they are eaten. Their sprite should become eyes. They can come out again after a certain period.
  - Refer to google pacman
  - The ghost's machine state is in the next Slide.

Advance (6%)Power Bean (2%)

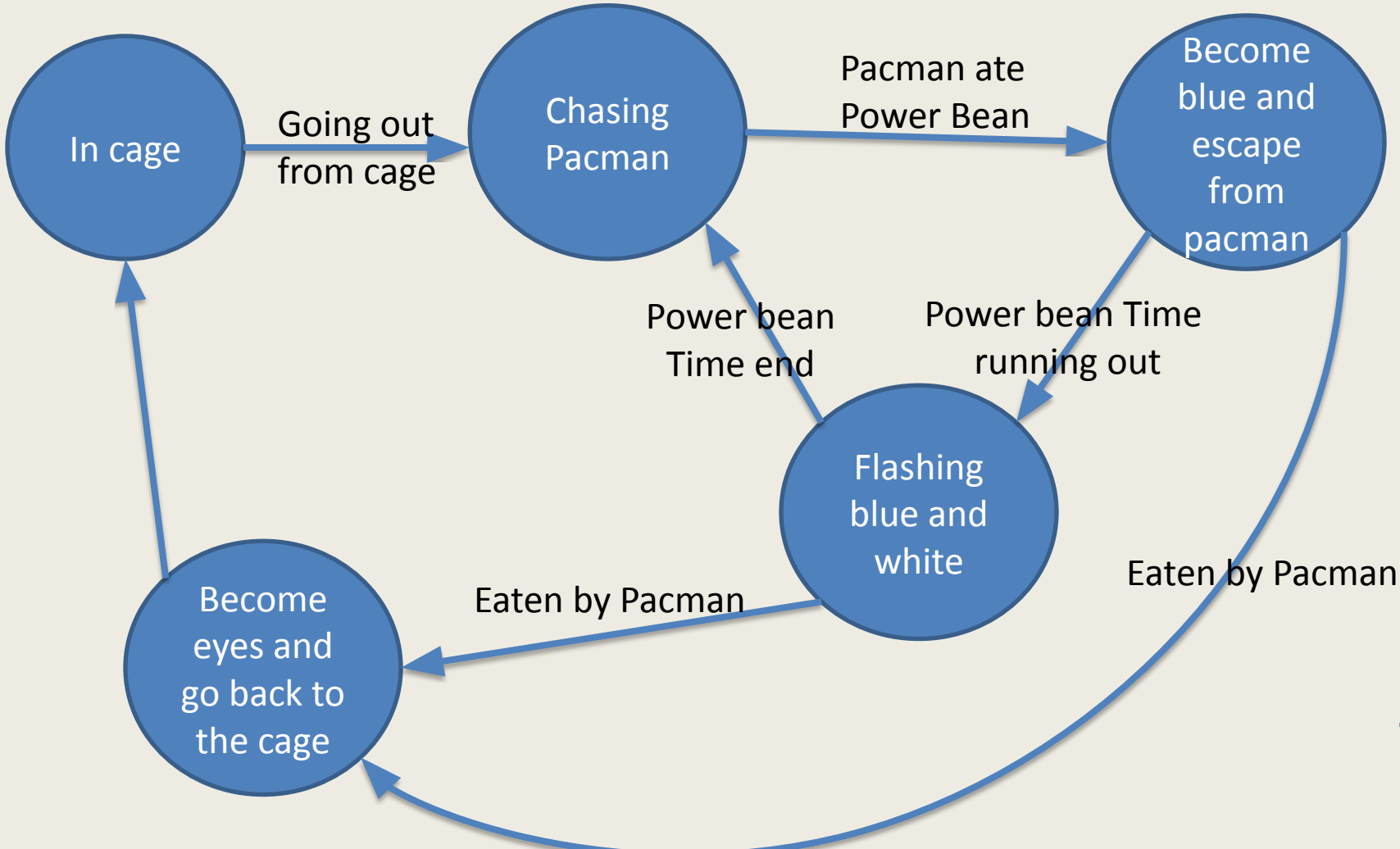
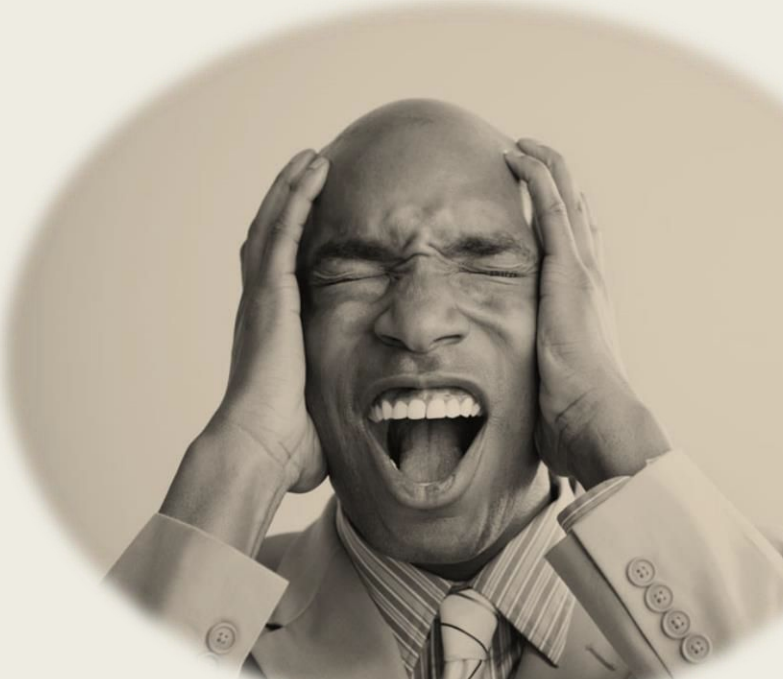


Image Reference

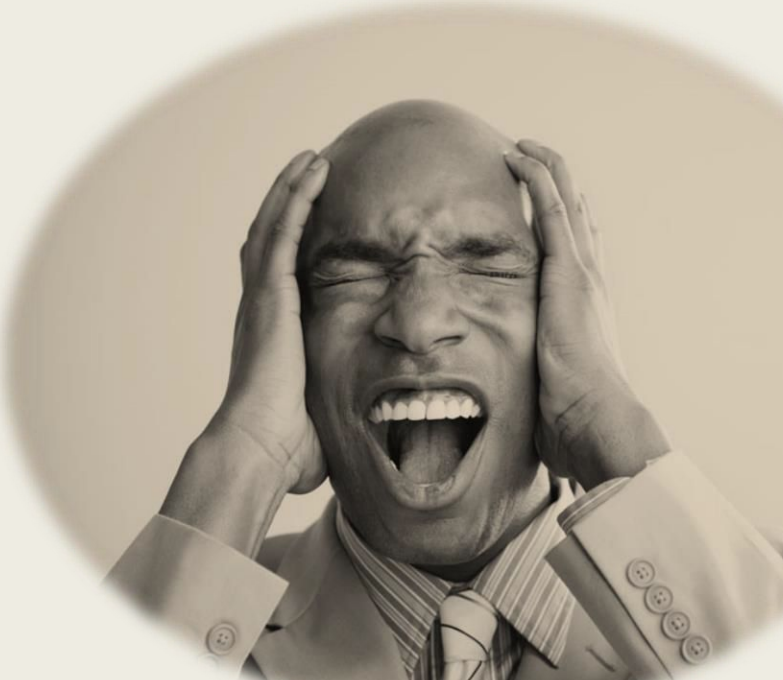
Advance (6%) Gameplay(1%)

- Pacman animation
- Ghost movement animation
- Sprites are provided in **Assets** folder
- You can use your own sprites, too
  - Four direction, each direction with at least two different sprites.



Advance (6%) Art(1%)

- Different BGM in different scene.
- Different version sound effects.
- Beautify UI
- Beautify Map
- Others related to Art.
- (Pick any two above)
  - (You should describe “Others”during the demo by yourself.)



Advance (6%) Gameplay(1%)

- Design another two items and design its effect. (MUST be reasonable)
- Reasonable Effect
  - Speeding up
  - Ability of passing through walls.
  - Activate Portal
- Unreasonable effect
  - +50 points
  - +game seconds

Note: Implementing one of the two blocks (green and blue) is enough.

- Map choosing or multi-level games
- Multiplayer (2P collaborating)

Advance (6%) Function、Interface Part(1%)

- Choose characters
- High score table
  Record the score and have a list of score records.

Both of them are required.

Creative (2%)

- Character appearance
- Magnificence attack
- Cool animations
- Richness of your game
- Good Performance (no lags)
- .....
- Any other you think that it’s hard to implement or special.
  - Implement them and list them at demo.

Template

- Multiple file template
  - *Template .zip*
  - functions and scenes are separated to different files.

Template (if you use Allegro 5.0)

- Change

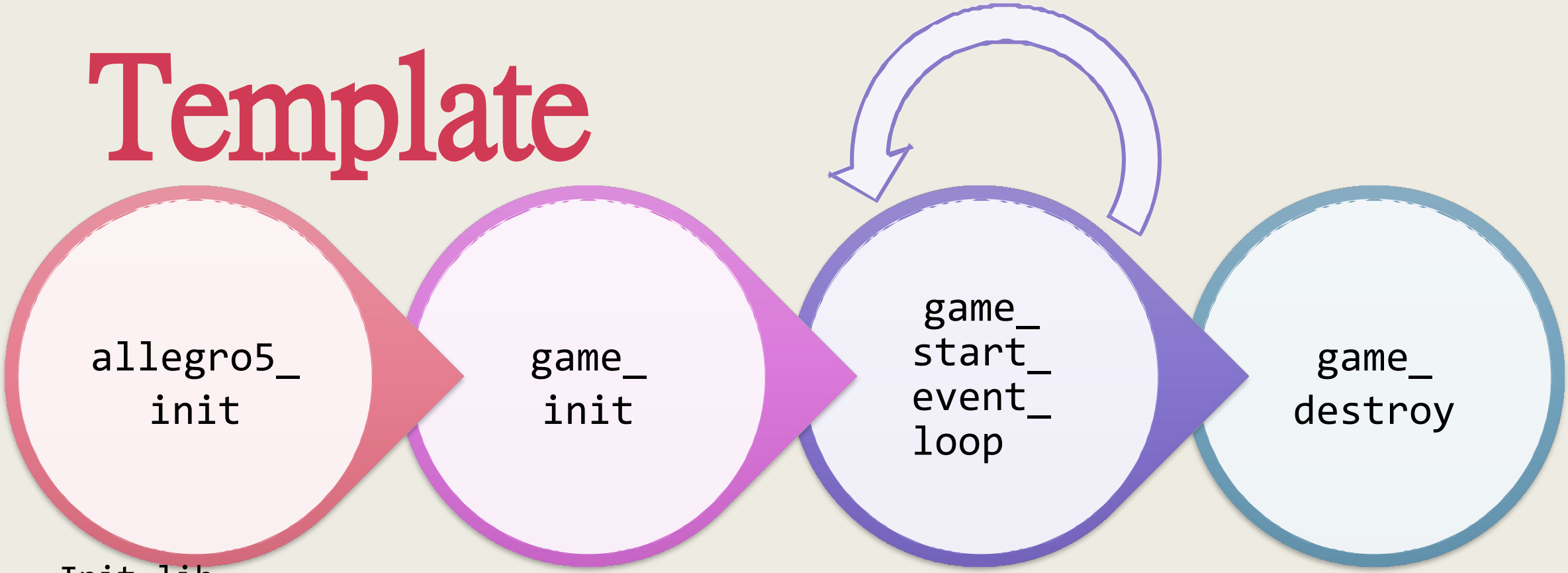
```
if (!al_init_font_addon())
    game_abort("failed to initialize font add-on");
```

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

- 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 object {
    Pair_IntInt Coord; //
    Pair_IntInt Size; // x for width, y for height
    Directions facing;
    Directions preMove;
    Directions nextTryMove;
    uint32_t moveCD;        // movement CountDown
} object;
```

Template(enum)

```
typedef enum Directions{
    NONE = 0,     UP = 1,
    LEFT = 2,     RIGHT = 3,
    DOWN = 4,     UP_DOWN = 5,
    LEFT_RIGHT = 6,     UP_LEFT = 7,
    DOWN_LEFT = 8,     DOWN_RIGHT = 9,
    UP_RIGHT = 10
} Directions;
```

Template(struct)

```
typedef struct RecArea{
    float x, y, w, h;
} RecArea;
typedef struct Pair_IntInt {
    int x;
    int y;
} Pair_IntInt;
```

```
typedef struct bitmapdata{
    int bitmap_x;
    int bitmap_y;
    int bitmap_w;
    int bitmap_h;
} bitmapdata;
```

Template(structs)

```
typedef struct Pacman{
    bitmapdata imgdata;
    object objData;
    func_ptr move;
    int speed;
    bool powerUp;
    ALLEGRO_TIMER* death_anim_counter;
    ALLEGRO_BITMAP* move_sprite;
    ALLEGRO_BITMAP* die_sprite;
} Pacman;
```

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 infinite loop.
 void game_start_event_loop(void);
// Release resources.
void game_destroy(void);
// Function to change from one scene to another.
```

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)

```
// 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 (draw)

```
RecArea drawArea = getDrawArea(pman->objData, GAME_TICK_CD);

    //Draw default image
    al_draw_scaled_bitmap(pman->move_sprite, 0, 0,
        16, 16,
        drawArea.x + fix_draw_pixel_offset_x,
    drawArea.y + fix_draw_pixel_offset_y,
        draw_region, draw_region, 0
    );
```

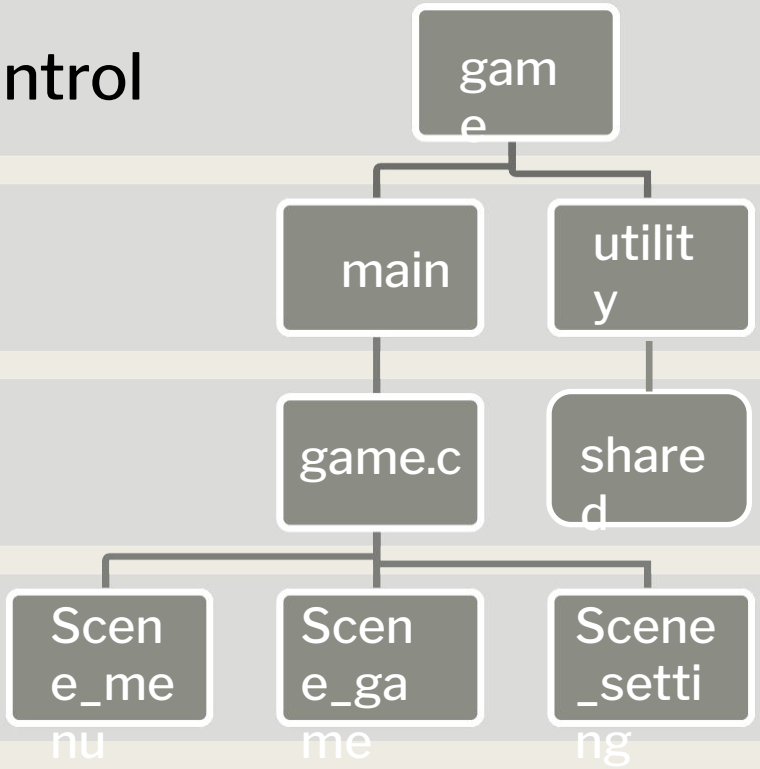
Template Structure

Allegro5 routines & Scene control

Utility functions & Entry point

Shared resources

Scenes



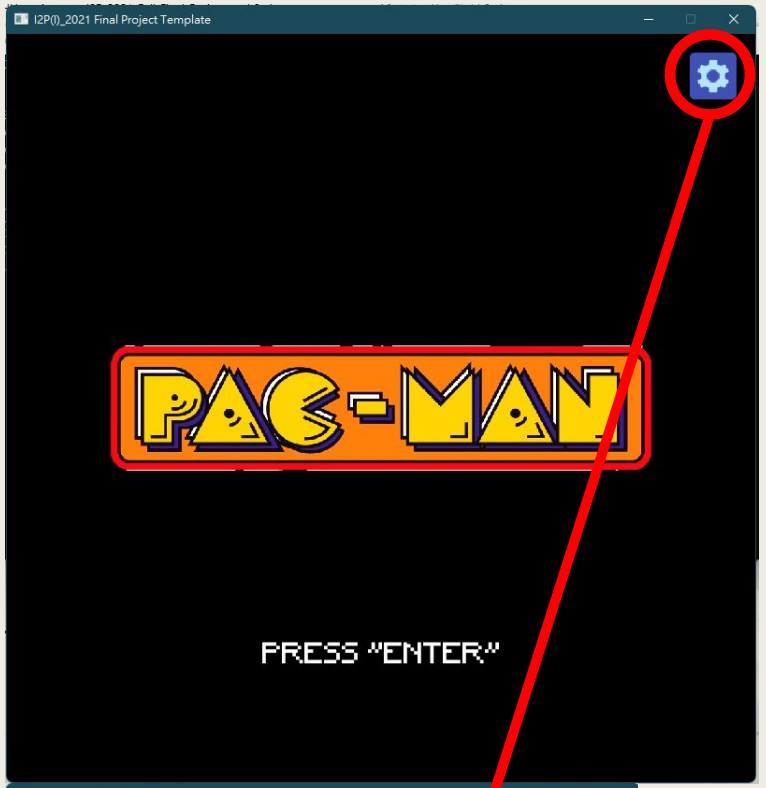
Today’s Goal

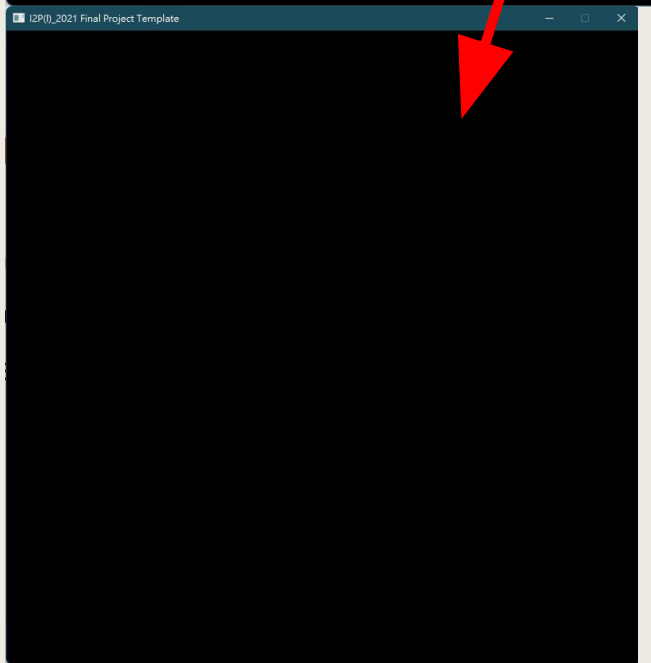
- Pacman Movement and Eat Bean
- Ghost Go Out & random movement (may go back and forth)
- Mouse event(Click) and enter setting scene

  - *Create the settings scene. (can be entirely black with no functions)*
  - *A button in main scene. (w/*

Today’ s Goal

- Pacman Movement and Eat Bean
- Ghost Go Out & random movement (may go back and forth)
- Mouse event(Click) and enter setting scene
  - *Create the settings scene. (can be entirely black with no functions)*
  - *A button in main scene. (w/ mouse in/out*





Today’s Goal (Example)

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

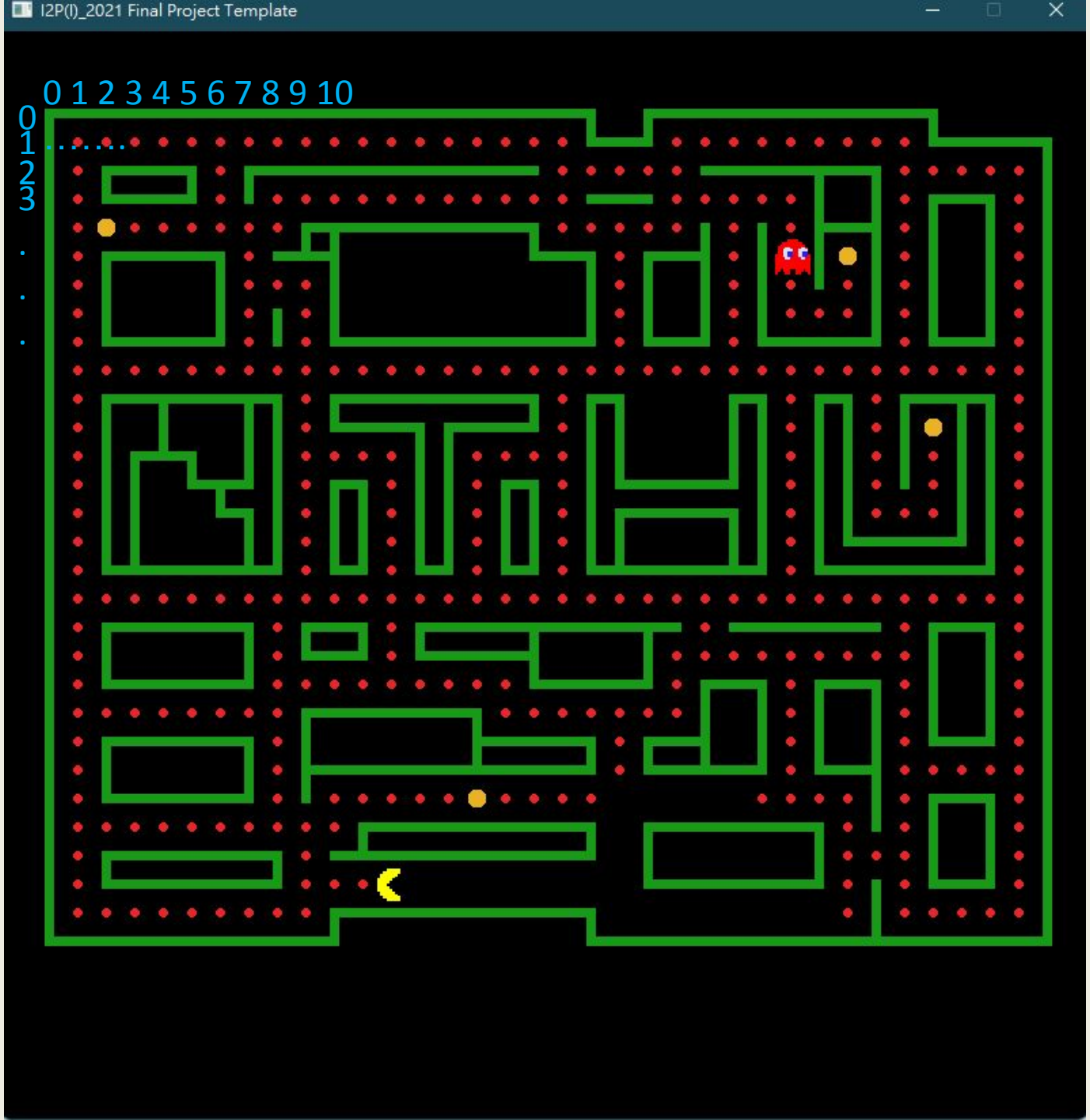
```
// [HACKATHON 1-1]
        // TODO: Use allegro pre-defined enum ALLEGRO_KEY_<KEYNAME> to controll
pacman movement
        // we provided you a function `pacman_NextMove` to set the pacman's next
move direction.
        /*
        case ALLEGRO_KEY_W:
            pacman_NextMove(pman, ...);
            break;
        case ALLEGRO_KEY_A:
            pacman_NextMove(pman, ...);
            break;
        case ALLEGRO_KEY_S:
            pacman_NextMove(pman, ...);
            break;
      .........
```

Today’s Goal (I)

- Setup movement for your pacman
- (HACKATHON 0-1) line 161 in map.c for loading map
- [HACKATHON] 1-1 ~ 1-4
- Separate the x and y axes. Use the same calculation to detect each axis

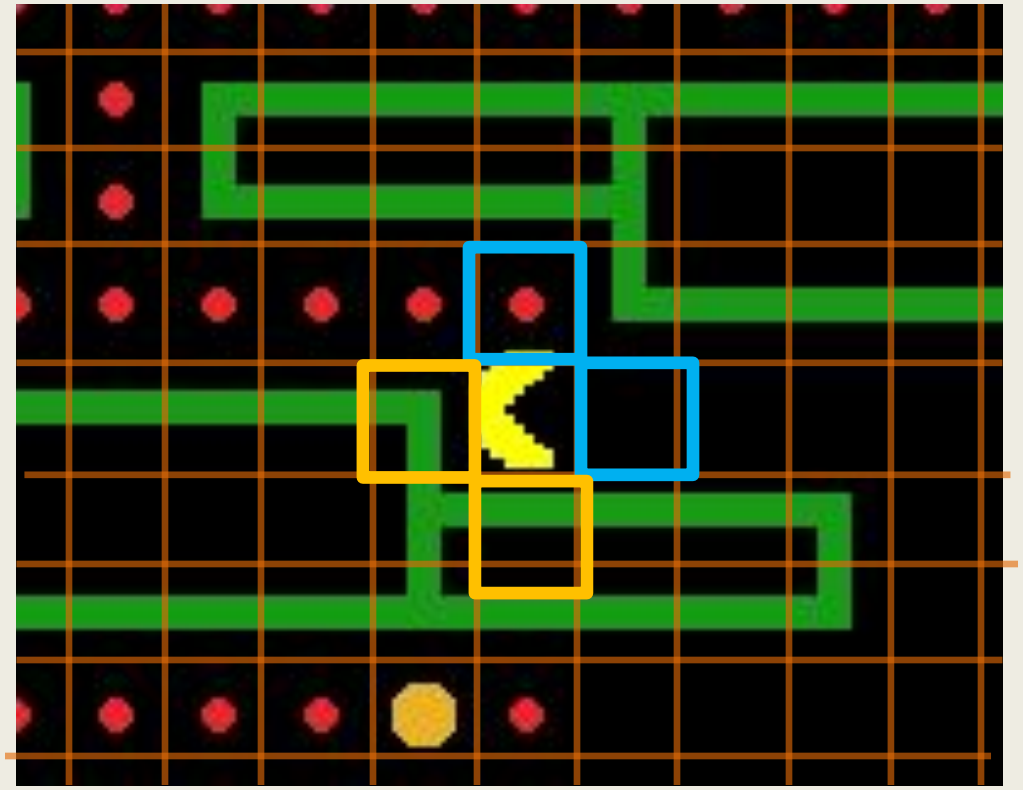
```
// [HACKATHON 0-1]
            // You can just switch to nthu_map if you want to finish HACKATHON 0 later.
                M->map[i][j] = default_map[i][j];
            //  M->map[i][j] = nthu_map[i][j];
```

Today's Goal (I)



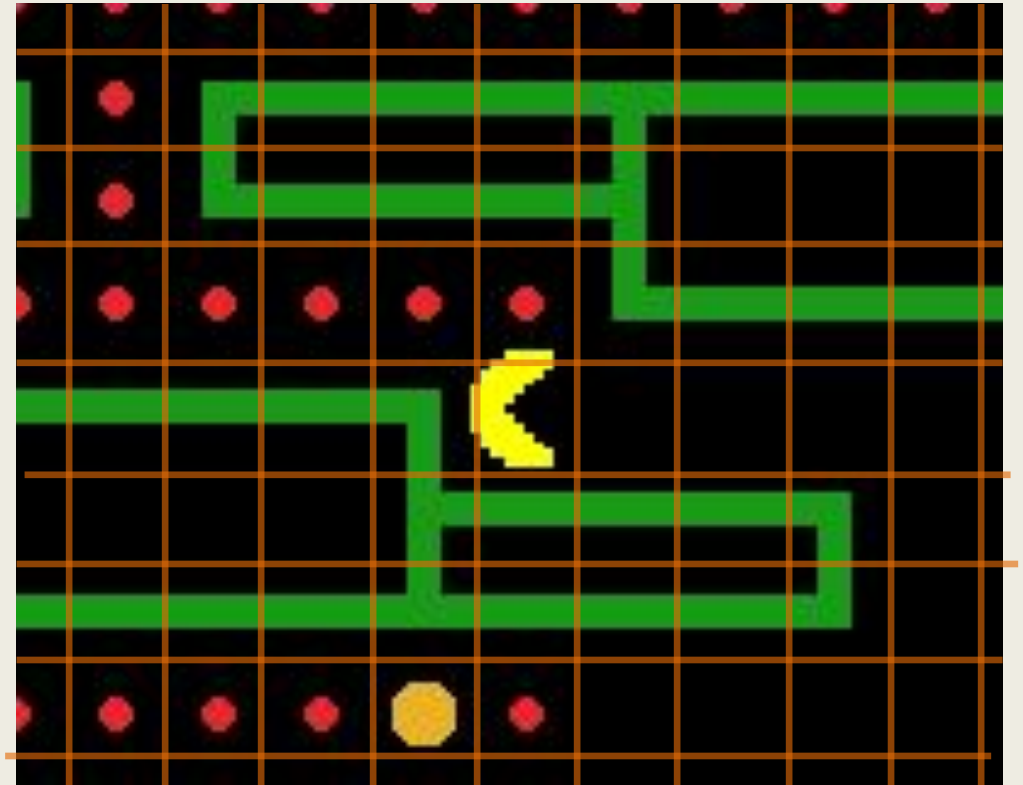
Today’s Goal (I)

- [HACKATHON 1-2] Setup Check of valid movement in pacman_movable(...)
- Valid
- Non-Valid



Today’ s Goal (I)

- [HACKATHON 1-3~4] Use `pacman_eatItem(...)` to activate the effect of item. (Playing sound)
- And erase the item from 2-D char Array map.



Today’s Goal (II)

- Allocate ghosts. (Today, one ghost is enough.)
- Let Ghost start to move.
- [HACKATHON] 2-0 ~ 2-4
- Control the state of ghost
- `ghost_movable` use the same logic of `pacman_movable`
- Today, only focus on the `ghost_red_move_script_FREEDOM` function.
  - But the state machine of ghost movement is important for your future programming.

```
typedef enum {
    BLOCKED,
    GO_OUT,
    FREEDOM,
    GO_IN,
    FLEE
} GhostStatus;
```

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)*
- [HACKATHON] 3-1 ~ 3-9

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

- 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 3 goals to get 3% score.
- The TA will ask you to explain how the 3 goals are implemented, you’ll get 3% score if you can describe how the code works.
- (each goal deserves 1% score respectively)

胡

Useful Resource

- **Allegro 5 Wiki**
  - https://www.allegro.cc/manual/5/
- **Allegro 5 reference manual**
  - https://liballeg.org/a5docs/trunk/
- **Movie Tutorial**
  - https://www.youtube.com/watch?v=IZ2krJ8Ls2A&list=PL6B459AAE1642C8B4
- **2D Game Development Course**
  - http://fixbyproximity.com/2d-game-development-course/

DON’ T CHEAT

LET’S CODE

Have a nice day~