COMP110: Principles of Computing

Software Testing

Today's lecture

Today's lecture has **three parts**

- Software testing and test-driven development
- Introducing COMP110 Coding Task II
- Object composition in C++

FALMOUTH
UNIVERSITY

The main loop

Basic program architecture

- CPUs execute sequences of instructions

The basic main loop

The most basic main game loop does **three** things:

1. Handle **input**
   - Mouse, keyboard, joypad etc.
   - Operating system events (minimise, close, alt+tab etc.)
2. **Update** the state of the game
   - Physics, collision detection, AI etc.
3. **Render** the game to the screen

It does these **once per frame** (typically 30 or 60 times per second)

The basic main loop

```
bool running = true;

while (running)
{
    handleInput();
    update();
    render();
}
```

Handling input

There are two ways of handling input in a game:

- By handling events
  - `SDL_PollEvent`
  - See `https://wiki.libsdl.org/SDL_EventType` for a list of event types
- By querying state
  - `SDL_GetKeyboardState`, `SDL_GetMouseState`, `SDL_GameControllerGetAxis`, etc.

What's the difference?

- Event: "The space bar was (pressed / released)"
- State: "The space bar is (down / up) right now"

Updating the game state

- Generally this is where your **game logic** is implemented
- I.e. anything not directly related to input or graphics
- What goes in here depends on the game...

Rendering

- This is where you draw the **current state of the game** to the screen
- Also draw any **heads-up display (HUD)** elements, e.g. score, lives, mini-map, etc.
- Graphical effects (animations, particles) may be handled **either** in the render step **or** in the update step (but be consistent)
- In frameworks like SDL, you generally **redraw everything** on every frame
- Rendering in SDL is **double buffered**
  - `SDL_Render*` actually draws to an **off-screen buffer**
  - `SDL_RenderPresent` displays the off-screen buffer on screen

Screen refresh rate

- Old CRT monitors worked by scanning an electron beam down the screen
  - `https://www.youtube.com/watch?v=lRidfW_l4vs`
- Hence the term **(vertical) refresh rate**
- Refresh rate is measured in **cycles per second** i.e. **Hz**
- Other monitor technologies work differently, but still refresh the screen at regular intervals
- We generally want to sync it up so that

one display refresh = one main loop iteration

- If the main loop runs too slowly, we get "lag"
- If the main loop runs too quickly, we waste resources on drawing things faster than the display can show them

Limiting the frame rate

- If the renderer was created with the `SDL_RENDERER_PRESENTVSYNC` flag, `SDL_RenderPresent` waits for the next vertical blank
- This limits the game's frame rate to the refresh rate of the device
- However, refresh rates can vary
  - Older TVs: $\sim$ 30Hz
  - HDTVs and standard monitors: 60Hz
  - High-end "gaming" monitors: 120Hz or higher

Socrative `6E8NSW3IN`

Why might updating the game state once per frame be undesirable?

- In pairs.
- Discuss for 2-minutes.
- **Suggest** an undesirable effect that might result from updating the game state exactly once per frame.

Measuring elapsed time

```
bool running = true;
Uint32 lastTime = SDL_GetTicks();

while (running)
{
    Uint32 currentTime = SDL_GetTicks();
    Uint32 deltaTime = currentTime - lastTime;
    // deltaTime is the number of milliseconds since ↩
        the last update

    handleInput();
    update(deltaTime);
    render(deltaTime);

    lastTime = currentTime;
}
```