Basic gameplay programming.

Part 2

Mouse Input

Mouse motion

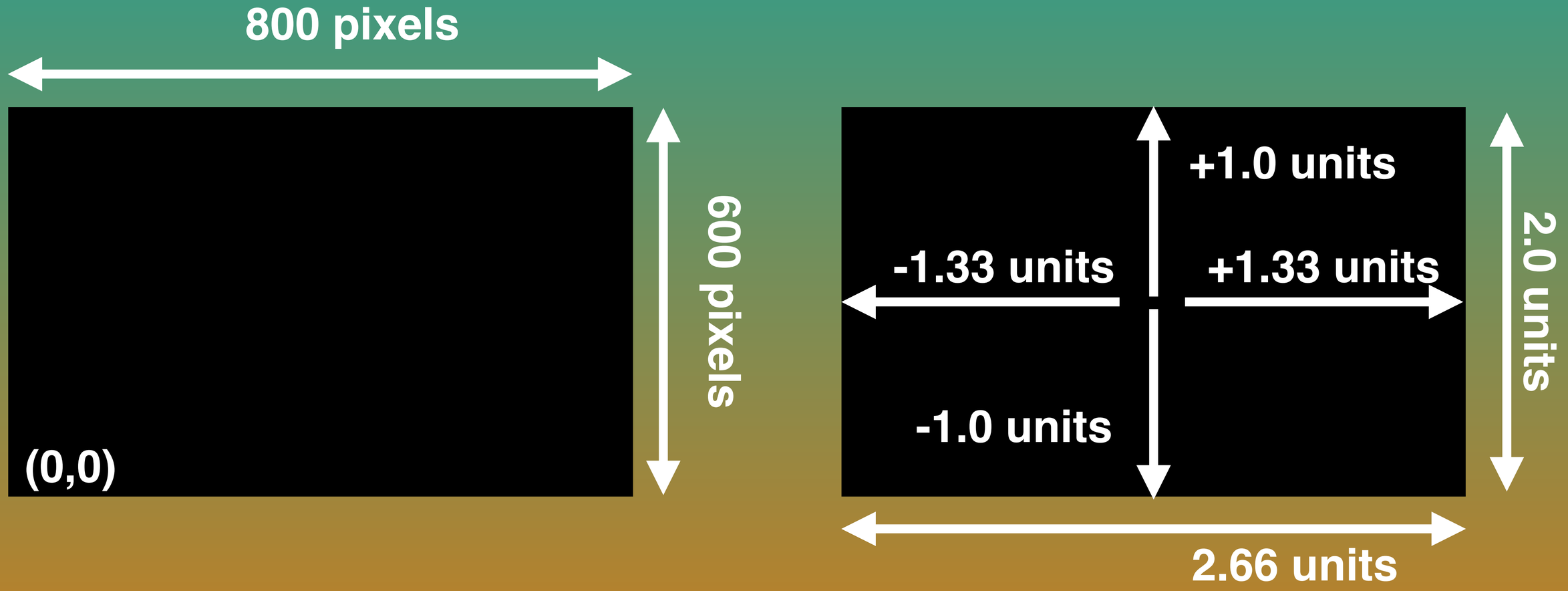
To respond to mouse motion, we must listen for the SDL_MOUSEMOTION event. We can then check the new position of the mouse by using the event.motion.x and event.motion.y variables.

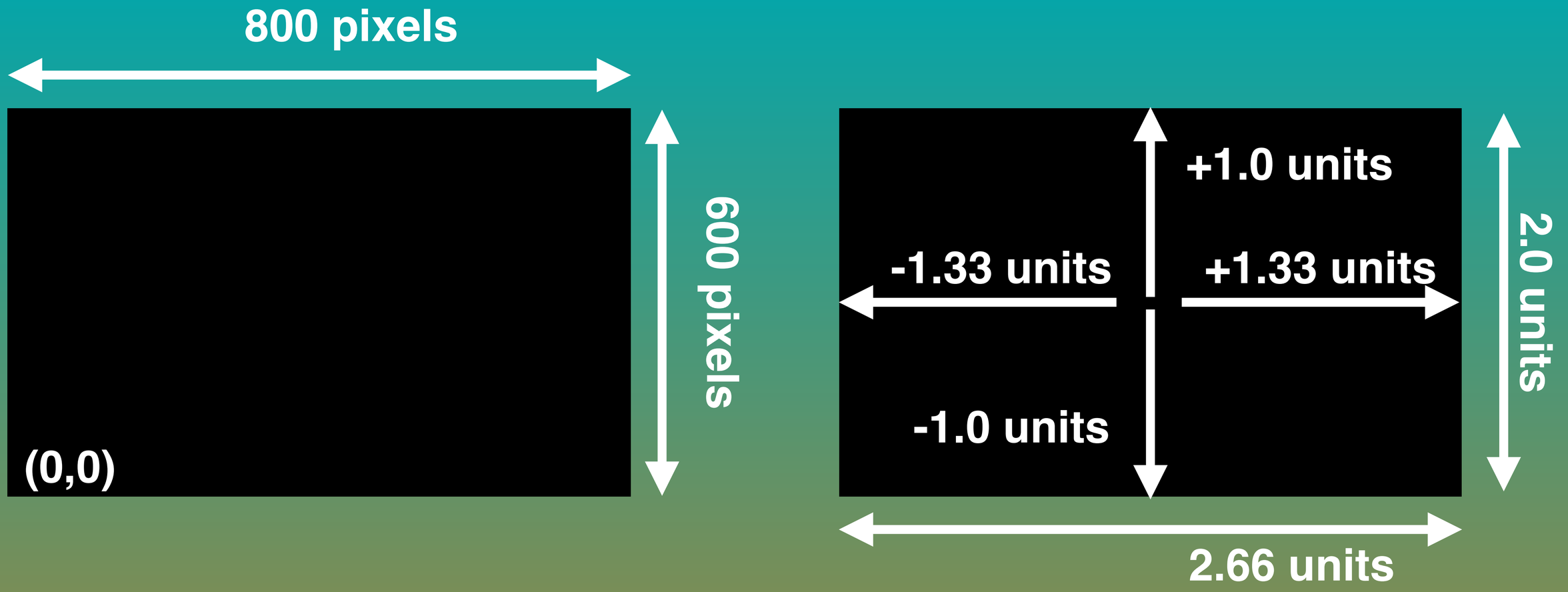
```
while (SDL_PollEvent(&event)) {
    if (event.type == SDL_QUIT || event.type == SDL_WINDOWEVENT_CLOSE) {
        done = true;
    } else if(event.type == SDL_MOUSEMOTION) {
        // event.motion.x is the new x position
        // event.motion.y is the new y position
    }
}
```

Converting from pixel coordinates to OpenGL units.

glViewport(0, 0, 800, 600);

glOrtho(-1.33, 1.33, -1.0, 1.0, -1.0, 1.0);





UNITS_X = (**PIXEL_X** / **X_RESOLUTION**) * **OPENGL_WIDTH**) - **OPENGL_WIDTH** / **2.0**;

UNITS_Y = ((**Y_RESOLUTION** - **PIXEL_Y**) / **Y_RESOLUTION**) * **OPENGL_HEIGHT**) - **OPENGL_HEIGHT** / **2.0**;

Converting from pixel coordinates to OpenGL units.

UNITS_X = (**PIXEL_X** / **X_RESOLUTION**) * **OPENGL_WIDTH**) - **OPENGL_WIDTH** / **2.0**;

UNITS_Y = ((**Y_RESOLUTION** - **PIXEL_Y**) / **Y_RESOLUTION**) * **OPENGL_HEIGHT**) - **OPENGL_HEIGHT** / **2.0**;

```
while (SDL_PollEvent(&event)) {
    if (event.type == SDL_QUIT || event.type == SDL_WINDOWEVENT_CLOSE) {
        done = true;
    } else if(event.type == SDL_MOUSEMOTION) {

        float unitX = (((float)event.motion.x / 800.0f) * 2.666f ) - 1.333f;
        float unitY = (((float)(600-event.motion.y) / 600.0f) * 2.0f ) - 1.0f;

    }
}
```

Mouse clicks

To respond to mouse clicks, we must listen for the SDL_MOUSEBUTTONDOWN and/or SDL_MOUSEBUTTONUP (for mouse release) events. We can then check which mouse button was clicked using event.button.button (1, 2, 3, etc.) and the position of the click using event.button.x and event.button.y variables.

```
while (SDL_PollEvent(&event)) {
    if (event.type == SDL_QUIT || event.type == SDL_WINDOWEVENT_CLOSE) {
        done = true;
    } else if(event.type == SDL_MOUSEBUTTONDOWN) {
        // event.button.x is the click x position
        // event.button.y is the click y position
        // event.button.button is the mouse button that was clicked (1,2,3,etc.)
    }
}
```

Using controllers

Using controllers

```
SDL_Init(SDL_INIT_VIDEO | SDL_INIT_JOYSTICK);

// SDL_JoystickOpen is passed the joystick index

SDL_Joystick * playerOneController = SDL_JoystickOpen(0);

// game loop

// clean up for each open joystick

SDL_JoystickClose( playerOneController );
```

Controller axis motion

To respond to controller axis motion, we must listen for the SDL_JOYAXISMOTION event. We can check which axis is moved by looking at the event.jaxis.axis variable and the new value of the axis using the event.jaxis.value variable.
event.jaxis.which tells us which controller this event is for.

```
while (SDL_PollEvent(&event)) {
    if (event.type == SDL_QUIT || event.type == SDL_WINDOWEVENT_CLOSE) {
        done = true;
    } else if(event.type == SDL_JOYAXISMOTION) {
      // event.jaxis.which tells us which controller (e.g. 0,1,etc.)
      // event.jaxis.axis tells us which axis moved (0 for x-axis ,1 for y, etc.)
      // event.jaxis.value tells us the new value of the axis from -32767 to 32767
    }
}
```

Controller button presses

To respond to controller buttons, we must listen for the SDL_JOYBUTTONDOWN and/or SDL_JOYBUTTONUP (for button release) events. We can check which button was pressed by looking at event.jbutton.button variable.

event.jbutton.which tells us which controller this event is for.

```
while (SDL_PollEvent(&event)) {
    if (event.type == SDL_QUIT || event.type == SDL_WINDOWEVENT_CLOSE) {
        done = true;
    } else if(event.type == SDL_JOYBUTTONDOWN) {
      // event.jbutton.which tells us which controller (e.g. 0,1,etc.)
      // event.jbutton.button tells us which button was pressed (0,1,2…etc)
    }
}
```

Organizing our code

```
void main() {

    Setup();

    while(loop) {
        ProcessEvents();
        Update();
        Render();
    }

    Cleanup();
}
```

```
void main() {

    Setup();

    while(loop) {
        ProcessEvents();
        Update();
        Render();
    }

    Cleanup();
}
```

```
void Setup() {

    // setup SDL
    // setup OpenGL
    // Set our projection matrix
}
```

```
void main() {

    Setup();

    while(loop) {
        ProcessEvents();
        Update();
        Render();
    }

    Cleanup();
}
```

```
void Setup() {

    // setup SDL
    // setup OpenGL
    // Set our projection matrix
}

void ProcessEvents() {
    // our SDL event loop
    // check input events
}
```

```
void main() {

    Setup();

    while(loop) {
        ProcessEvents();
        Update();
        Render();
    }

    Cleanup();
}
```

```
void Setup() {

    // setup SDL
    // setup OpenGL
    // Set our projection matrix
}

void ProcessEvents() {
    // our SDL event loop
    // check input events
}

void Update() {
    // move stuff and check for collisions
}
```

```
void main() {

    Setup();

    while(loop) {
        ProcessEvents();
        Update();
        Render();
    }

    Cleanup();
}
```

```
void Setup() {

    // setup SDL
    // setup OpenGL
    // Set our projection matrix
}

void ProcessEvents() {
    // our SDL event loop
    // check input events
}

void Update() {
    // move stuff and check for collisions
}

void Render() {
    // for all game elements
    // setup transforms, render sprites
}
```

```
void main() {

    Setup();

    while(loop) {
        ProcessEvents();
        Update();
        Render();
    }

    Cleanup();
}
```

```
void Setup() {

    // setup SDL
    // setup OpenGL
    // Set our projection matrix
}

void ProcessEvents() {
    // our SDL event loop
    // check input events
}

void Update() {
    // move stuff and check for collisions
}

void Render() {
    // for all game elements
    // setup transforms, render sprites
}

void Cleanup() {
    // cleanup joysticks, textures, etc.
}
```

Entities

```
class Entity {
    public:

        void Draw();

        float x;
        float y;
        float rotation;

        int textureID;

        float width;
        float height;

        float speed;
        float direction_x;
        float direction_y;
};
```

Entities are a useful way for us to think about objects in the game.

Pong.

Assignment

- Make PONG!
- Doesn’t need to keep score.
- But it must detect player wins.
- Can use images or basic shapes.
- Can use keyboard, mouse or joystick.