Day 1: Hello Odin

Dylan <dylan@programvideogames.com> Mon, Jun 24, 2024 at 7:21 AM
To: brian.w.woodard@gmail.com

Hello there! Welcome to Program Video Games first mini-course!

By the end of this mini course you will have learnt how to program Pong from scratch.

That means you'll know:

- Basics of the Odin programming language
- How to draw shapes to the screen
- How to handle input
- How to program moving and colliding objects
- How to program simple AI

By the end of this lesson, you'll know:

- How to get up and running with Odin
- How to create your first program

Why Pong?

A *lot* of games are made using the exact same principles that you'll learn by making Pong.

They are just combined in ways that make the games more complex.

If you peek behind the curtain of a bunch of games, even new ones, you'll find a lot of it is moving rectangles colliding with each other.

What is (game) programming?

Game programming is known to be extremely difficult.

I'm going to show you how to break it down into manageable chunks.

Like all computer programs, games are made up of cycles of just 3 steps: Input, Processing, Output.

Here's a simple example of how a player moving in a game is just Input→Processing→Output.

Input: The player moving the thumbstick + the player's position and speed

Processing: Add the speed of the player to the player's position

Output: Draw the player in the new position

These three steps are repeated at different scales at every level of the program.

Want to add two numbers together? Input: 420, 69

Processing: 420 + 69

Output: 489

The whole game itself is a set of inputs that interact with the game's state (what level is loaded? what enemies are there nearby? etc) to produce the output which is shown to you as the game on your screen.

Our tools: Odin, Raylib

Odin is a simple and clean programming language that is suited for games due to it's speed. It is comparable to C and C++, which are industry standard languages for game programming.

Why not use C or C++? There's a lot of cruft and things that get in the way of learning game programming when using these languages.

Rest assured, anything you learn here will be transferrable to those languages if you choose to switch later.

Raylib is a simple and easy-to-use library to enjoy videogames programming.

It comes with much functionality that you'd have to reproduce by hand otherwise.

The way it's structured makes it simpler to understand than many alternatives.

Installation

Odin + Raylib

Raylib is shipped with Odin, so we don't need another install step.

Windows

For windows users, I have a PowerShell script that will install everything you need into `C:\ProgramVideoGames`.

You can download it here.

Once downloaded, right click the file and select Run with PowerShell.

When it's done, open the Command Prompt (cmd) and type in `C:\ProgramVideoGames\BuildTools\devcmd.bat` and hit enter.

This will load all the tools we installed so they can be accessed in the terminal.

Test it out by typing `odin version`. You should get something like this: `odin version dev-2024-06-nightly:f745a1c47`.

We'll use this window to compile and run our programs.

MacOS and Linux

Follow the instructions on this page: https://odin-lang.org/docs/install/

```
package main

import "core:fmt"

main :: proc() {
    fmt.println("Hello there!")
}
```

- `package main` - every file in Odin must have a package declared. For most projects you'll use the same package and by convention that's main. Every file in the same folder must have the same package name.
- `import "core:fmt"` - `import` keyword allows you to pull in code from other package. The fmt package is included in the core resource that comes with Odin.
- `main :: proc()` - functions in Odin take the name "procedure" hence the `proc` keyword. Your program always starts in main.
- `fmt.println` - use the println procedure from the fmt package. It prints text to the terminal.

Run the program with `odin run hello.odin -file`.

You should see "Hello there!" in the terminal.

Now let's look at this program through the lens of Input → Processing → Output.

Zoomed out view

Input: The hello.odin file

Processing: Odin compiler parses the file

Output: hello.exe - a working program

Main procedure view

Input: Nothing

Process: Call fmt.Println procedure

Output: Nothing OR printed text

I put this distinction here because our main procedure doesn't return a value. It just calls another procedure which causes a side-effect of printing text out.

Println view

Input: "Hello there!"

Processing: Code inside println procedure

Output: Nothing OR printed text

Even with this tiny program there are 3 or more (if you look inside println) instances of Input → Processing → Output.

Join me in the next lesson where we'll get a window up and running, handle keyboard input, and draw a rectangle that moves when you press buttons!

If you have any questions, make sure to check out the Discord server.