GAME DEVELOPMENT

2018 Progress

Course notes, projects, research

P1xt[1]

[1] https://github.com/P1xt

These are my notes, musings, solutions, research and anything else I could think to keep track of for all courses, research, projects, etc I'm working through as part of learning game development with Rountable Interactive.

KEYWORDS

Game Development, Mathematics, Algorithms, Computer Science

Contents

Part I

Courses

CSCI-E-23a - Introduction to Game Development

This course picks up where Harvard College's CS50 leaves off, focusing on the development of 2D and 3D interactive games. Students explore the design of such childhood games as Super Mario Bros., Legend of Zelda, and Portal in a quest to understand how video games themselves are implemented. Via lectures and hands-on projects, the course explores principles of 2D and 3D graphics, animation, sound, and collision detection using frameworks like Unity and LÖVE 2D, as well as languages like Lua and C#. By class's end, students will have programmed several of their own games and gained a thorough understanding of the basics of game design and development.

Course Site: `https://cs50.github.io/games/`

This is Harvard's followup course to CS50 for Games Development. It's not yet available on edX but the lectures and assignments are available online so I'm following along. Languages: lua and C#(Unity).

Assignment 0 - Pong

Highlights

- Lua
- LÖVE2D
- Basic OOP (Object-Oriented Programming)
- Drawing Shapes
- Drawing Text
- DeltaTime and Velocity
- Game State
- Box Collision (Hitboxes)
- Sound Effects (with bfxr)

Lecture

TABLE 1 Important Links

Purpose	Link
Installing love2d	`https://love2d.org\#download`
Running love2d	`https://love2d.org/wiki/Getting\_Started`
Demo code	`https://github.com/games50/pong`
Game Programming Patterns (online book)	`https://gameprogrammingpatterns.com`
bfxr (create audio effects)	`https://www.bfxr.net/`

Important Concepts

Definition Game Loop

The intention of the Game Loop pattern is to decouple the progression of game time from user input and processor speed. It processes input, updates the game, and then renders anything that has changed in the game's state. The important takeaway is that while it processes input, it doesn't wait for it. It is responsible for running the game at a consistent speed, regardless of where it's running (old pc, android device, etc).

Definition 2D Coordinate System

Typical cooridinate system, but with the origin (x_0, y_0) at the top left of the system. Positive - to the right and down. Negative - to the left and up.

Definition Class

A blueprint for encapsulating methods and attributes which allow you to instantiate objects which follow that blueprint. Each object created has it's own data. It is convention to capitalize class names. The ":init" method is used as a constructor. "self" is used in a class definition to reference the current object's instance variables.
The advantage of Object Oriented programming with regards to games development is that it allows us to defer update and render logic to each object, keeping our main code much shorter and more maintainable.

Definition AABB Collision Detection
Relies on colliding entities to not be rotated (have "axis-aligned bounding boxes" and enables the use of a simple formula to detect collisions between them.

```
if a.x is not > b.x + b.width and
   a.x + a.width is not < b.x and
   a.y is not > b.y + b.height and
   a.y + a.height is not < b.y:
   collision = true
else
   collision = false
```

For more info see: `https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection`

Definition State Machine Monitors the current state of the game, what states are possible, and what transitions take place to effect a change from one state to another.
Note: later look up Pushdown Automation `https://en.wikipedia.org/wiki/Pushdown_automaton` which employs a stack of states so that you can not only know the current state, but also prior states.

TABLE 2 Important Functions - Love2D

Function	Description
love.audio.newSource(path, [type])	creates an audio object which can be played at any point in the program. Type can be "stream" or "static" - static is held in memory, stream is read from disk.
love.draw()	renders any changes to the screen
love.event.quit()	terminates the game
love.graphics.clear(r, g, b, a)	fills the entire screen with the desired color
love.graphics.newFont(path, size)	loads a font into memory
love.graphics.printf(text, x, y, [width], [align])	prints text to the screen, can align text left, right or center
love.graphics.rectangle(mode, x, y, width, height)	draws a rectangle to the screen, mode: fill or line
love.graphics.setColor(r, g, b, a)	sets the currently active color for drawing shapes or text
love.graphics.setDefaultFilter(min, mag)	texture scaling filter - bilinear(blurry), nearest(retro)
love.graphics.setFont(font)	sets the currently active font
love.keypressed(key)	callback, executes when key pressed
love.keyboard.isDown(key)	returns true or false depending on whether the specific key is currently being pressed
love.load()	initializes game state
love.resize(width, height)	Called by Love2d when we resize the application. Any code that needs to be run if the window is resized (like push:resize() goes here)
love.timer.getFPS()	returns the current FPS of the application
love.update(dt)	called each frame / dt is elapsed time since last frame
love.window.setMode(width, height, params)	window init, typically use push library instead
love.window.setTitle(title)	sets the title of the application window

TABLE 3 Important Functions - Other

Function	Description
push.setupScreen(VW, VH, WW, WH, params)	initialize virtual resolution, params table: fullscreen, resizable, vsync
push.apply("start")	begin rendering at virtual resolution
push.apply("end")	end rendering at virtual resolution
math.randomseed(num)	seeds the random number generator used by math.random
math.random(min, max)	returns a random number between min and max
math.min(num1, num2)	returns the lesser of num1 and num 2
math.max(num1, num2)	returns the greater of num1 and num 2
os.time()	returns the the time since Jan 1, 1970 in seconds

Assignment

The assignment for Pong was pretty simple - add an AI player to control one or both of the paddles. The way I tackled this was to add a few more attributes to the paddles to make them more generic (like what controls a paddle uses, and whether it's controlled manually or by automated), then handle movement based on the state of those variables. Snippets:

```
    -- player 1
    if player1.autoplay == true then
        autoPlay(player1)
    else
        processPlayerInput(player1)
    end

    -- player 2
    if player2.autoplay == true then
        autoPlay(player2)
    else
        processPlayerInput(player2)
    end
```

```
--[[
    A function that will cause the passed player's Paddle
    to be be moved according to player input
]]
function processPlayerInput(player)
    if love.keyboard.isDown(player.upkey) then
        player.dy = -PADDLE_SPEED
    elseif love.keyboard.isDown(player.downkey) then
        player.dy = PADDLE_SPEED
    else
        player.dy = 0
    end
end

--[[
    A function that will cause the control of the passed player's
    paddle to be automated
]]
function autoPlay(player)
    if ball.y > player.y + player.height / 4 then
        player.dy = PADDLE_SPEED
    end
    if ball.y < player.y  - player.height / 4 then
        player.dy = -PADDLE_SPEED
    end
    if ball.y > player.y and ball.y < player.y + player.height then
        player.dy = 0
    end
end
```

Assignment 1 - Flappy Bird

Highlights

- Images (Sprites)
- Infinite Scrolling
- “Games are Illusions”
- Procedural Generation
- State Machines
- Music
- Mouse Input

Lecture

Assignment

18.01 - Single Variable Calculus

This calculus course covers differentiation and integration of functions of one variable, and concludes with a brief discussion of infinite series. Calculus is fundamental to many scientific disciplines including physics, engineering, and economics.

Unit 1: Differentiation

Part A: Definition and Basic Rules

Geometric Interpretation of differentiation - find the tangent line to $y = f(x)$ at $\underset{\sim}{P} = (x_0, y_0)$.

Definition $f'(x_0)$, the derivative of f at x_0, is the slope of the tangent line to $y = f(x)$ at $\underset{\sim}{P}$.

Definition Tangent Line is the limit of secant lines PQ as $Q \to P$ where P is fixed.

FIGURE 1 Main Formula

$$f'(x_0) = \lim_{x \to 0} \frac{f(x_0) + \Delta x) - f(x_0)}{\Delta x}$$

Problem Set 1

Part II

Projects

Part III

Books

Part IV

Research