Justin Dietrich, Ryan Scott

11/20/2022

CST - 310

Prof. Citro

Project 8

Simple Demo Scene

**Mathematical Concepts:**

Pong runs off a series of complex translations on the X and Y axis. The paddle will move up and down, and the ball will move in 8 directions. In order to bounce the ball off the paddles, some angle math is required:

GLfloat a = pi\*120\*(ballPosY - paddle2Y)/360;

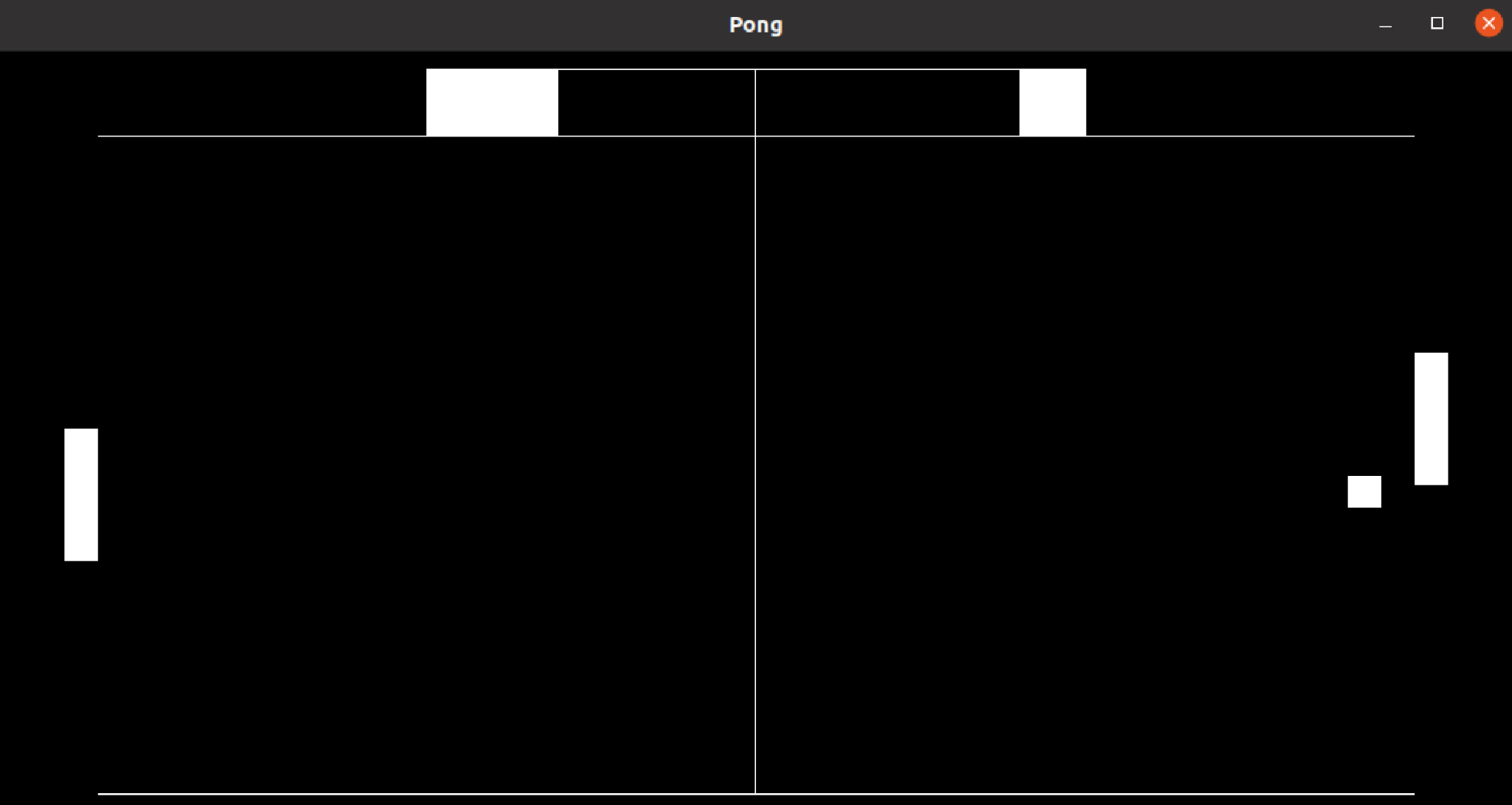
ballVelX = -cos(a) \* ballSpeed;

ballVelY = sin(a) \* ballSpeed;

**Screenshots:**

****

**Score: 0 - 0**

****

**Score: 2 - 1**

**Short video:**

<https://www.loom.com/share/2972fabc018b47e29cf20fe8712a2f05>

**Explains Why It Is Interesting to Watch, Challenging to Design, and Tricky to Program:**

As one of the earliest video games in history, Pong was the first electronic game that attracted wide public attention when computers were capable of handling such raw creative energy. At the time, it was the most interesting thing humans could think of to put on a screen. Due to the different edge cases that must be addressed in order to achieve Pong's unique buttery smooth gameplay, its design can be challenging. These include:

* Collision with left paddle: let it fly to the right and set the angle (or y-direction) depending on where it hit the paddle.
* Collision with right paddle: same as left paddle, just to the opposite direction.
* Collision with left wall: increment the right player's score and reset the ball to the center.
* Collision with right wall: increment the left player's score and reset the ball to the center.
* Collision with top wall: invert the ball's y-direction.
* Collision with bottom wall: invert the ball's y-direction.
* Increase speed after every ball hit

**Controls:**

Left Paddle:

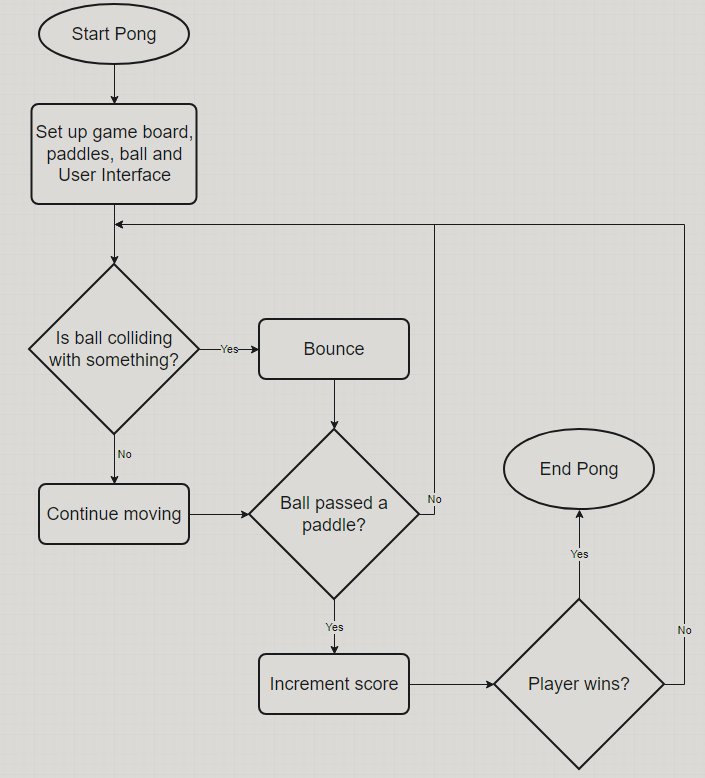
|  |  |
| --- | --- |
| **Command** | **Hotkey** |
| Move up | W |
| Move down | S |

Right Paddle:

|  |  |
| --- | --- |
| **Command** | **Hotkey** |
| Move up | U |
| Move down | J |

The game will stop when a player reaches 5 points and you can restart it with ‘R’.

**Flowchart:**

****

**Theoretical Background, Main Idea, Various Components, Concepts, Relationships, and Intended Effect:**

With the use of translations and simple trigonometry, we were able to create a beautiful, simplistic game of Pong that would appeal to the eye. Unlike in Unity, which allows us to attach rigid bodies and apply velocity to GameObjects, all of the work must be done by hand, and these relationships must be translated manually. Using enough if statements and clever trigonometry, we are able to recreate the smooth bouncy movement we are accustomed to.

**Programming Implementation:**

Since we were majoring in game design, we figured that object-oriented programming would be the most appropriate approach. In OpenGL, classes are the closest we can get to this, so we made one for the paddles and ball. The paddles move up and down using a simple translation function. During the code generation, the ball is analyzed to determine whether it bounces on top or bottom, and if it bounces, the Y velocity is flipped. Depending on the angle at which the ball collided with the paddle (using pi and some simple cos and sin math), the ball bounced and its X velocity is flipped.

**Appropriateness:**

Pong is not only moral, but ethical and legal too!

**Sources:**

ASCII Table - ASCII Character Codes, HTML, Octal, Hex, Decimal. (n.d.). <https://www.asciitable.com/>