LabVIEW Pong

Team Members

- 109550194 龍偉亮
- 109550193 關盛文

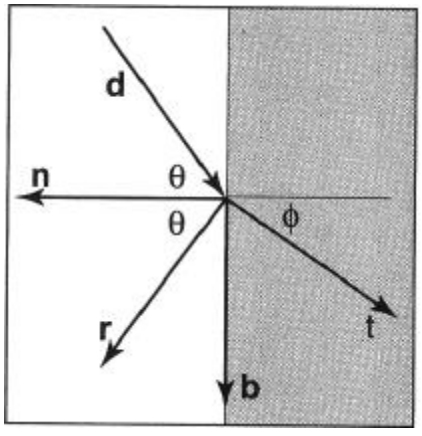
Motivation

Our motivation was to test if we could make a classic video game from scratch in LabVIEW and Pong came to mind. Pong is a 1 to 2 player game and if we could succeed in training a model to play our game, we could make the game play automatically on its own, or even surpass ourselves in playing.

Design

A standard Pong game consists of a ball and one player on each side of the screen. There are designs that could have the players move in 2d space but we decided to have ours move up and down in 1d space first as a prototype. One reason being that model training would be simpler with binary output. We wanted to have a working framework first.

The ball would move in one direction, and reflect as it hits the ceiling, floor, or the players. If it reaches the end of either side, left or right of the screen, one of the player gets one point. The reflection of the ball is calculated with the following equation,



$$r = d - 2\,(d \cdot n)\,n$$

The ball has its starting direction randomized in a fixed range to avoid softlock which would happen if the ball direction is vertical. And the closer it is to vertical, the slower the game progresses, but also not too close to horizontal which would make the ball not

utilize enough of the playing space if it only bounces left and right. But also, it's just not fun that way.

We designed it so that a starting dialog would let the players set a winning score, which when either player reaches that target score wins and the game ends. The game pauses for every new round when a player scores. And if the game ends, another dialog pops up so that players can decide on whether to quit the game or set another target score and restart the game.

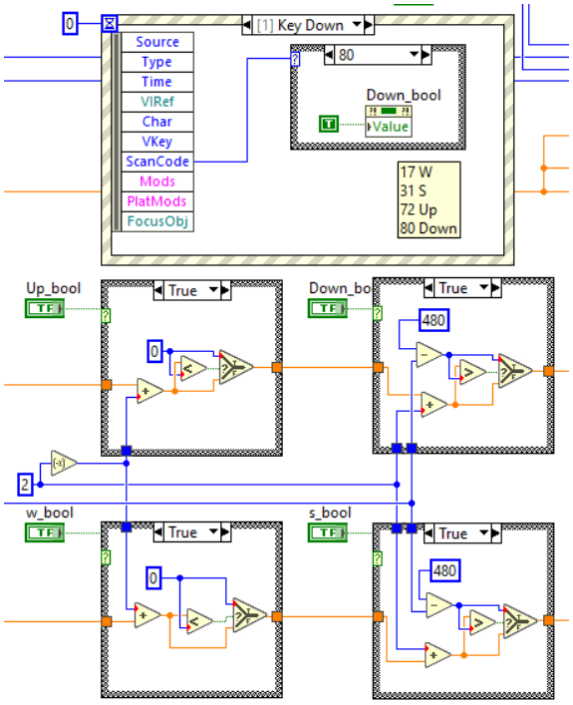
Implementation

The ball needs a vector of x and y to keep track of its moving direction, then its speed, and its position. Its radius defines how big the ball should be.

The players have their controls mapped to physical keys so that we can control the players on our input devices. And the players have positions to keep track of where each player is, and player width and height to draw the player boxes.

As for the player movements, at first we tried the Event Structure to handle Key Repeat. But as we found out the moment when a key is held down, and another key gets pressed, the event of the first Key Repeat gets cancelled by the second key press. So in order to get around this with our own implementation to let both players move freely by holding down keys without having to repeatedly mash their buttons, we used the Key Down and Key Up events to simulate holding down keys, which sets their corresponding key Booleans to true or false.

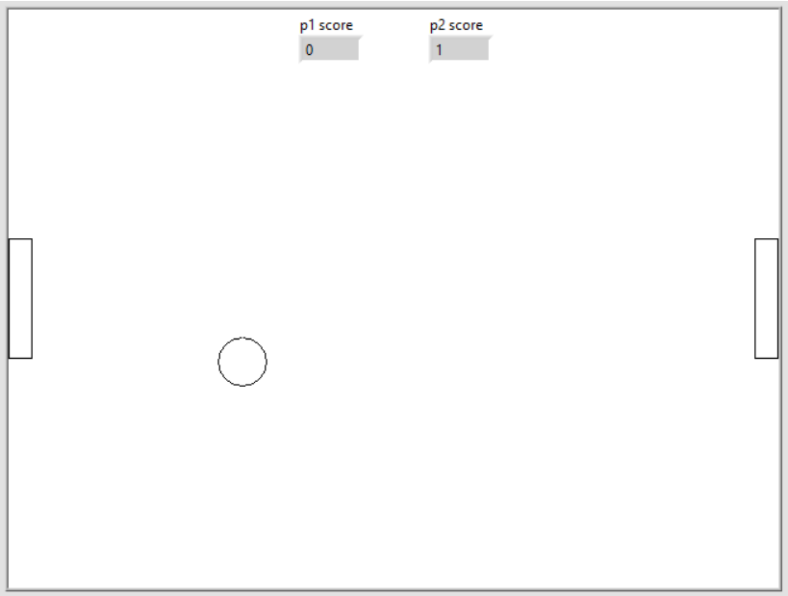
Then several case structures below the event structure will handle the calculations.



A pause button lets players pause the game, but we made it so that only the ball is frozen and players could still move around because it’s fun.

Modifiers like player height, ball speed are implemented to change up the game difficulty and pacing.

We use picture control to “render” the game, we use a round circle to represent the ball, rectangles to represent the players. The scores are numerical indicators because they get the job done. The window size is fixed to 640x480.



Work Distribution

龍偉亮 is in charge of the main development of Pong in LabVIEW.

關盛文 is in charge of the model training and making it work with LabVIEW.

Design choices, ideas, problems met, etc. are frequently discussed.