Mini Project: Flappy Bird Interim Report

Shriya Singh & Jenny Nguyen
Group 31

Game Strategy

Flappy Bird is an engaging game that puts players in control of guiding a bird through a series of obstacles by managing its vertical movement. To navigate through the obstacles and avoid crashing into them or the ground, players utilise the left mouse button to make the bird ascend. Successfully manoeuvring through obstacles awards players with points, with the objective being to achieve the highest score possible.

The game offers two modes: Training and Normal. In Normal Mode, players face the traditional Flappy Bird experience, where they have only one life to progress through the obstacles. As players advance and increase their points, the game adjusts the speed of the obstacles, creating a more demanding and intense gameplay experience. Any collision with an obstacle or the ground results in an immediate game-over.

Training Mode fosters a more forgiving environment for players to improve their skills. Players will have access to three lives, so they can make mistakes without facing immediate failure. Each collision with an obstacle deducts one life, allowing players to learn and improve their performance over time. However, colliding with the ground, regardless of remaining lives, terminates the game.

While the scoring system remains active in both modes, Training Mode does not incorporate difficulty scaling. This allows players to familiarise themselves with the game mechanics and obstacles without the added pressure of increasing difficulty levels.

Design Specification

This project has defined design specifications. The game interface must utilise a VGA interface, constraining the available screen size, to 640x480 pixels. Additionally, maintaining a consistent refresh rate of 60 frames per second is important, which means we have to have synchronisation with a VGA clock period of 16.67 milliseconds.

Furthermore, the game must be compatible with the DE0-CV platform, which requires implementation using VHDL. It will also incorporate a four-state finite state machine, with the states being the "start_game", followed by “training mode”, “normal mode”, and the "game_over" states.

Plans

We require several components for our game: a bird component whose Y-position is controlled by mouse clicks and is capable of detecting collisions with pipes, ground and gifts to manage the game state and score.

A pseudo-random number generator (implemented with a LFSR) will generate the location of the gap in the pipes. Upon the collision of the bird with the pipe, a signal will be passed to the FSM. In the case the bird passes the pipe another signal will be sent to the score to increment it by 1.

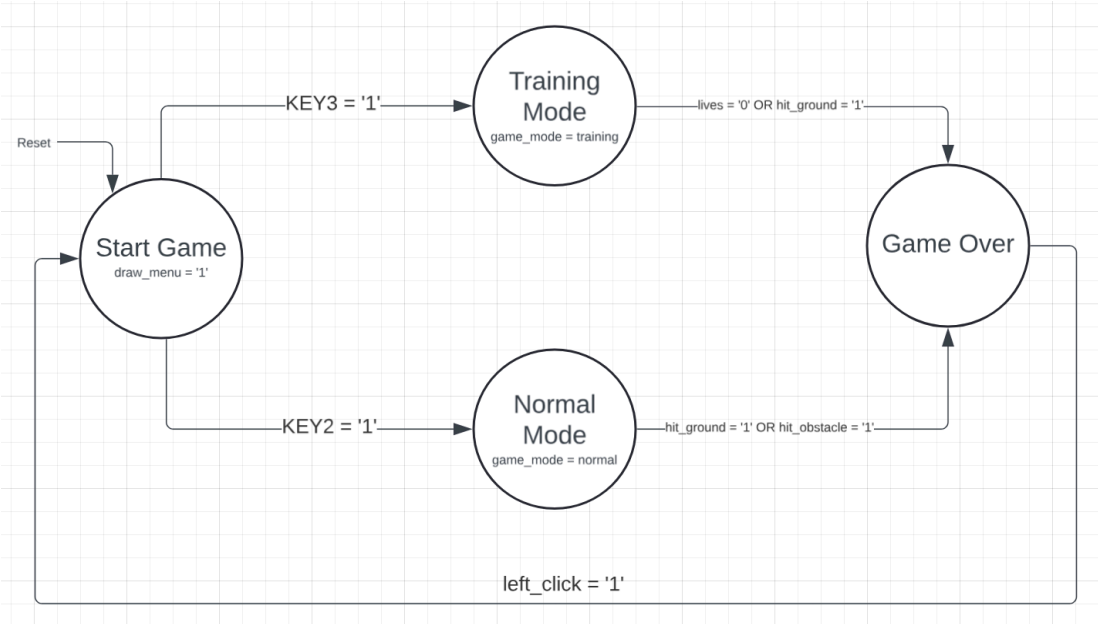
A display text component is responsible for displaying all the text on the screen like score and level. It uses the char_rom as a component to do this. We will be further implementing the game start screen and modes text using this as well.

The background has its own component that has an output to the VGA_sync. Our pipes will also be its own component that will be displayed on the screen. Each component being displayed on the screen has a priority order that determines what will be displayed on top of the background, pipes etc.

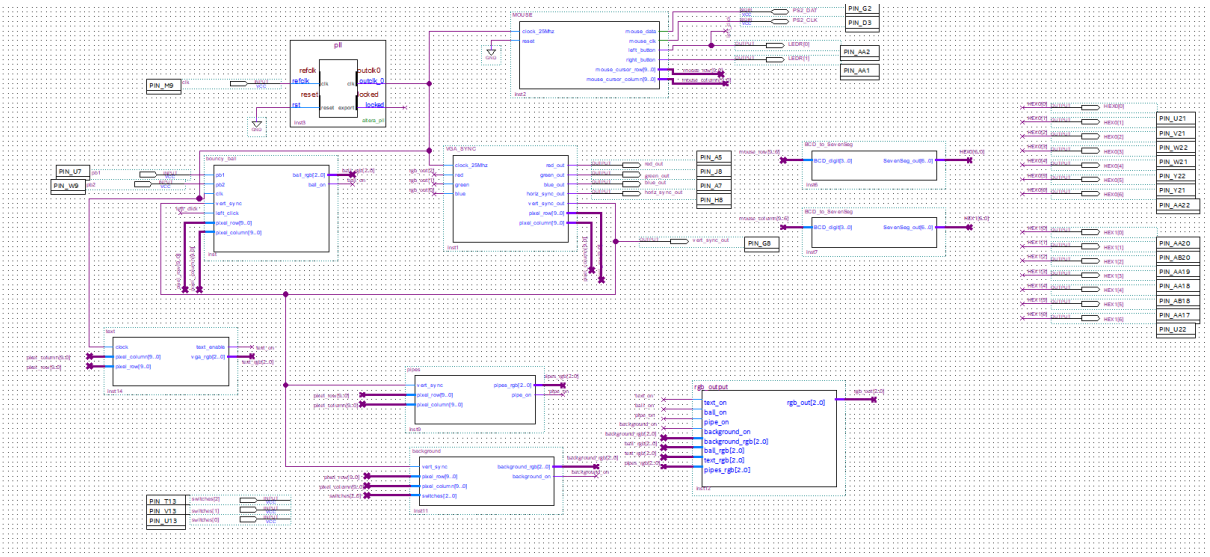
A gift component will be implemented that allows the player to increase their score by 2 whenever they collide with it. This will pass a signal to the score to increment this score by 2.

A Finite State Machine will be implemented to keep track of what part of the game the player is currently on. We are planning to create a four-state Moore machine that allows you to select the type of mode the player would like to play with, with the inputs of bird collision and key selection to change the states.

High-Level State Machine



Block Diagram



Block Diagram: Refer to Quartus for a larger diagram