Final Project: TETRIS

Fall 2019

Overview

Introduction

Welcome to the final stretch of ECE 550. In groups of two, you will complete a final project to show off your ECE/CS skills. Your project this year is going to be your favourite game - Fortnite. Just kidding, it's your good old classic *TETRIS*. Your project should make substantial use of the provided processor (to be released with documentation after the processor deadline). You can modify and create any hardware that you like and you will be writing software (in the form of our ISA) to make use of that hardware.

Grading Overview

- Implementation (85%)
    - Baseline Game (80% of 85% = 68% overall)
    - Add-on Features (20% of 85% = 17% overall)
- Demo/Presentation (15%)

Grading Policy

Your game will be graded in the following way: we will define "levels" of completion for each feature. If you have successfully completed a level, you will obtain a score between this level and the level below it based on how well you have implemented this feature (see the last page for a grading example). All TAs will discuss when deciding your grade to ensure fairness. Refer to the rest of this document for specifics.

Project Specification

Baseline (68%)

The biggest portion of your grade will come from a baseline TETRIS game described below. The main gameplay (moving blocks) must be displayed on a VGA screen to receive points.

I/O (12%)

Your game should include I/O for the players to control the movement of falling pieces.

- Level 1 6%: using the buttons/switches on the FPGA
- Level 2 12%: using PS2 keyboard

Falling Pieces of Different Shapes (17%)

Your game should have pieces of different shapes that fall at a rate of at least 2 Hz (i.e. refresh every 0.5 seconds) until it reaches the bottom.

- Level 1 9%: at least one shape with 4 or more squares
- Level 2 13%: at least three shapes with 4 or more squares; at least one of them can't be a rectangle
- Level 3 17%: at least five shapes with 4 or more squares; at least two of them can't be a rectangle

Clear/Manage Lines (17%)

Once a line is filled, all squares on this line should be cleared. If the top of the box (play area) is reached, the game will stop.

- Level 1 5%: the game stops if the top of the box is reached.
- Level 2 9%: line can be cleared but squares on higher lines do not drop to fill in the empty space
- Level 3 17%: line can be cleared and squares on higher lines can drop

Rotation (10%)

The player should be able to rotate the shapes in 90 degrees granularity.

- Level 1 5%: shapes can be rotated in 90 degrees granularity without using the processor
- Level 2 10%: shapes can be rotated in 90 degrees granularity using the processor

Scores (12%)

Your game should show the user a score for the current game.

- Level 1 7%: score is shown on the FPGA
- Level 2 12%: score is shown on the VGA screen

Using Processor: You have to use the provided processor to implement at least 2 tasks out of these 5 tasks. Otherwise, your implementation points will be deducted by 20%.

Add-ons (17%)

The rest of the implementation points come from your add-on features. You can implement any combinations of add-ons to get the 17%. As with the baseline points, your score will depend on the quality of your implementation, with the maximum score noted in parentheses next to each add-on option. Note that you cannot exceed 17% for add-ons. If you do attempt multiple add-ons, we will grade the best of the add-ons that add to 17% (again, see the grading example below for clarification).

Interrupts (12%)

Using processor (you will have to modify it), allow the user to interrupt the current game, e.g., the user can press a key to pause the game and resume, or to use a wild-card to eliminate all lines.

Controller (12%)

Build your own physical game controller and using GPIO on the FPGA to control movement and rotation.

Pseudo-Random Block Placement (12%)

Rather than hard-coding/pre-computing the placement of blocks in a new level, create a pseudo-random method of placing blocks in the level.

Leaderboard (5%)

Record scores for all games played in the current session and allow the user to display a leaderboard showing the top 3 players with some kind of identification (e.g., time).

Speed Power-Up (5%)

Cause the speed of block falling to increase/decrease as a result of a specific chain of events in your game (e.g. trigger a speed power-up after some type of game event).

Demo (15%)

Your demo will be comprised of four things:

- A 5 to 10 minute presentation
- A visual aid or deliverable
- A 5 to 10 minute Q&A session

The demo will be ~30 minutes. You must have some kind of deliverable to go along with your presentation — whether it be a Powerpoint, Prezi, video of your project, etc. — to facilitate your presentation. The deliverable should be such that any novice would be able to understand your entire project from it.

Your presentation will be graded based on your ability to articulate technical information in a concise and cohesive manner. Every group member must substantially participate in the presentation. Additionally, you will also be graded on how well you can answer our questions. Finally, you will also be graded on your deliverable.

Implementation Grading Example

A digital designer from Duke's Class of Renaissance, Mr. Leo Da Vinci, got the following grade:

Baseline

I/O: Leo implemented using PS2 to control the FPGA and he could use keys to control movement of the shapes instantly. **12%**
Falling Pieces: Leo implemented 3 shapes, but the squares in his shapes overlap with each other. He would get a score between 9 to 13% because he completed Level 2, but the TAs decided to give him **11%** because of the squares overlapping.
Clear/Manage Lines: In Leo's TETRIS, the game stops when the top of the play area is reached and lines can be cleared but squares on higher lines would not drop. In addition, sometimes a filled line will clear a neighboring line. He would get a score between 0 to 9% because he completed Level 1, but the TAs decided to give him **7%**.
Rotation: Leo didn't implement rotation. **0%**
Scores: Leo displayed a score on the right corner of the VGA screen that increases correctly with time. **12%**

Baseline Subtotal = 12% + 11% + 7% + 0% + 12% = 42%

Add-Ons

Interrupts: Leo modified the processor to cause the game to pause when he hits the "P" key on the keyboard. However, the pause key does not work when a player is moving blocks. He would get a score between 0 to 12% because he attempted interrupts, but the TAs decided to give him **8%**.
Pseudo-Random Block Placement: In Leo's game, different block layouts exist with different playthroughs. He used data from user input through the keyboard and a series of processor arithmetic operations to "randomly" layout blocks in his game. **12%**
Leaderboard: Leo tracks the scores for all games played in the current session and offers the option to display a leaderboard showing the top 3 scores, identified by the time at which the game with that score ended. **5%**

Add-ons Subtotal = Max(8%, 12%) + 5% = 12% + 5% = 17%

Implementation Score = Baseline + Add-ons = 42% + 17% = 59% (Out of 85%)