Amiel Iliesi

Mr. Winikka P1D1

9/9/16

Programming 2 / Applied Web Design

Tetris Senior Project (proposal and outline)

**Proposal:**

I am proposing a project to learn and recreate, rather than create a new idea or concept. My project idea is to recreate the iconic 2D puzzle game *Tetris* using Python (Pygame?) and OpenGL, as well as some yet-to-be-determined web services to upload my game (or just a private host on a computer for budgeting purposes; I don’t need the world to see my efforts, just those around me. Although it is an established game, it works as a learning tool to showcase my learning capabilities as well as expand my knowledge of the back-end of Python and OpenGL.

The game seems simple enough, but just as talking about something simple in your native language would be incredibly difficult in a new language, programming an established game would have me needing to learn a large sum of the back-end of Python. I would need to learn about a graphical interface, responsive inputs, ‘physical’ detection of blocks (hitboxes), sprite management, object grouping (lines can break sprites apart), general data management (points, speed, lines completed), state machines (for the menu), colorblind assistance, audio management (music and possibly SFX), and web connection (game access, highscore upload, contact, etc.). I plan to learn these things intermittently using various (yet to be determined) sources, while mainly operating in Python based off of my outline, with some OpenGL use as well (I have yet to know how/what OpenGL does but my project mentor Chris Winikka recommended the program implementation in my project so finding a place for that will also become a part of my research). Beyond programming and (G)UI, I will also have to make my graphics using various image processing utilities, as well as making the music (or finding/requesting the music with an appropriate liscence). The target platform will be an in-browser access to the game, it will function on Chrome, Edge, and Firefox (if I can make that happen).

I will approach my research by looking up beginner tutorials and attempting to self-teach whatever concepts I am able to. I am attempting to singlehandedly recreate this classing from the ground up, so as little assistance from both the internet and others will be my unofficial goal.

**Outline: *-NOTE- there are sub-outlines not specified here in my folder that deal with nuances***

1. The main structure of the program
   1. Game start
   2. Menu
      1. Play
      2. Options
         1. Music
         2. Colorblind mode (optional)
            1. Figure out how to do this
         3. Configure controls (optional)
      3. Help
         1. Tutorial
         2. Controls
         3. Highscores
         4. Credits / info
         5. Download button? (optional)
2. Figure out how to make objects in Python or JS, and how to display physical objects…
   1. If this is figured out, this all becomes possible, so get on it!
3. The game’s Boolean arithmetic, including–
   1. Each piece’s logic (**:: :.. ..: .:· ·:. .:.** )
      1. Also including its interactions with the physical playable area / border itself
   2. Row completion detection
   3. Ability to rotate detection
      1. Then assign the arithmetic for the rotation itself (if necessary)
   4. Game failed logic
   5. RNG overlord
   6. A piece “hold” function that complies with existing game logic
   7. Include a timing system that increases linearly
      1. Only increase timing after each piece is placed OR after a set number of pieces are placed
         1. Do NOT increase timing *during* a turn
   8. Ensure that the movement inputted into the blocks keeps the pieces on a grid
      1. If a diagonal button is pressed then either route the buttons to be pressed in order at tick speed OR use Pythagorean theorem to scale the movement to match the grid in a diagonal motion
      2. Pulse inputs, but repeat if held (with the exception of two buttons being constantly held) to avoid repeated diagonal movement (disorientating)
4. Scoring system
   1. Line removal score
      1. If 1 line removed in 1 turn, give 10 points
      2. If 2 lines removed in 1 turn, give 40 points
      3. If 3 lines removed in 1 turn, give 90 points
      4. If 4 lines removed in 1 turn, give 160 points
   2. Increasing downward motion points
      1. For every block sped up, give 1 point
      2. For every block when block is instantly placed below, give 2 points
   3. For every block placed, give 1 point (1 point per block, not block pieces; 1 point not 4)
5. Visuals
   1. Use either premade sprites, or physical objects, whichever taxes the computer less
   2. Layer each aspect of the game “board”
      1. HUD
      2. Background
      3. Border
      4. Playable area
   3. Allow the game “board” to be scalable / fullscreen? (optional)
   4. If there is a tutorial, decide on static slides, or mutable objects
6. Web connection and game upload, playable online w/ highscore upload function
   1. Host with WebMatrix? or find a better alternative
   2. Submit an upload request if the user chooses to upload their score
   3. Compile a list of the top 20 scorers
   4. If someone surpasses slot 20, then remove slot 20 and readjust list with the new element
   5. If 2 or more elements tie, then choose the score to succeed the others based on score date (requires timestamps for every score data entry)
7. Extras
   1. Music
      1. Official music preferable if CC license
   2. Splash / Intro
   3. Tutorial