Pitch B: Another way to study machine learning would be to try and find the most effective way to parse the data on screen into information about a task that an AI could use to complete that task. For the sake of this study, I would use an open-source implementation of Tetris as the novel task. Tetris is a suitable task for this study as it is both simple and includes a random element that will prevent the AI from simply finding an optimal series of keys to press rather than finding a method to play the game. An intuitive way to conduct this study would be to find a suitable open-source machine learning AI and refine methods to turn the current screen into workable data that the AI can understand. However, it would be more effective to gather the needed data from already published methods of automatically parsing the screen into usable data. For that reason, I will seek out published studies that could provide all the needed data or at least insight into how to begin the process of parsing screen data.

Source 1: The Game of Tetris in Machine Learning

- Simon Algorta and Özgür Şimşek. 2019. The game of Tetris in machine learning. arXiv:1905.01652v2 [cs.LG] 10 May 2019. 7 pages. https://arxiv.org/abs/1905.01652
    - Notably, this article has sections dedicated to algorithms and features, the structure of the decision environment, and open challenges.
    - This article covers handcrafted controllers, genetic algorithms, and reinforcement learning and how they have been used in Tetris machine learning
    - This document clarifies what implementation of Tetris is typically used for this line of research
    - This article cites many previous attempts at conquering Tetris using machine learning

This article describes the history of notable and documented machine learning AI built to play Tetris. More critically, this document describes the implementation of Tetris used and the state features given to each AI. This knowledge of how previous AI have been designed and how successful they were will likely be key to knowing where to start my research. Furthermore, this article mentions that a good way to reduce game length and, thus, research time is to reduce the size of the playing space so a game over will occur sooner.

Source 2: Using dual eye-tracking to unveil coordination and expertise in collaborative Tetris

- Patrick Jermann, Marc-Antoine Nüssli, Weifeng Li. 2010. Using dual eye-tracking to unveil coordination and expertise in collaborative Tetris. © 2010, the Authors. 9 pages. https://www.scienceopen.com/hosted-document?doi=10.14236/ewic/HCI2010.7
    - This research project used eye tracking on a pair of Tetris players
    - Sadly this article is more focused on the human element than the machine learning
    - The article defined three significant things for a player to look at
        - Their block
        - Their partner's block (irrelevant to my own work)
        - And the contour of the stack
        - It may be worth getting the AI to focus on only the block and the contour and disregard the rest of the screen
    - The equations for the machine learning used in the research is included near the end of the document in the section Automatic recognition of pair composition
        - Sadly the machine learning aspect was focused on understanding patterns in player interaction, not how they played the game successfully
    - Somewhat interestingly, the article also notes patterns in the parts of the game focused on by more successful players

Although not very directly applicable to my project, there are some potentially essential bits of information to be gleaned. The article breaks down what parts of the screen were most relevant to players and, thus, what parts would be most relevant to an AI given only the current game screen as input. It also broke down what moves the players had available and when they tended to use them most often. Lastly, this article contains observations about what movement patterns and focuses were most common among more experienced players. Although much of it will have to be adapted back to a single-player game of Tetris, the information in this document has its uses.

Source 3: General Video Game Playing

- John Levine, Clare Bates Congdon, Marc Ebner, Graham Kendall, Simon M. Lucas, Risto Miikkulainen, Tom Schaul, Tommy Thompson, Michael Mateas, Mike Preuss, Pieter Spronck, and Julian Togelius. 2013. General video game playing. Artificial and

Computational Intelligence in Games. Dagstuhl Follow-Ups, Volume 6, ISBN 978-3-939897-62-0.,,; pp. 77–83 Dagstuhl Publishing Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany . 7 pages. https://strathprints.strath.ac.uk/57217/

  - a programming language called GDL or Game Description Language mentioned in this article could be a helpful lead
    - That same paragraph also mentions Planning Domain Definition Language
  - This article cites a more formal definition of what a game is, this could be a helpful paradigm to build around for this project.
  - Significantly this article focuses on games with continuous activity, like Tetris, and not just turn-based games
  - This article states that the two main approaches to building a general game-playing AI is deliberative reasoning (e.g. minimax search) or policy learning (e.g. reinforcement learning).
  - The article makes mention of an Arcade Learning Environment (ALE), based on the games for the classic Atari 2600 games console, near the end of page 4
  - The two main AI approaches to investigating a new game are based on reinforcement learning and Monte Carlo Tree Search
  - This article mentions a companion paper that presents Video Game Description Language definitions for three diverse arcade games

This article seems like a potential gold mine of useful sources and ideas. The mentioned companion articles and programming languages tailored to describe the game state to an AI seem to be very promising leads. Furthermore, the style of AI that works with a “first-person perspective” on the game sounds similar to what I had imagined. Looking more into what this style entails, or better yet, finding examples of AI built for this proposed competition, would be a wise investment of time. Needless to say, I have some new leads for my research after this article.

Source 4: Towards a Video Game Description Language

- Marc Ebner, John Levine, Simon M. Lucas, Tom Schaul, Tommy Thompson, Julian Togelius, Michael Mateas, Mike Preuss, Pieter Spronck, and Julian Togelius. 2013.

Towards a video game description language. Artificial and Computational Intelligence in Games. Dagstuhl Follow-Ups, Volume 6, ISBN 978-3-939897-62-0.; pp. 85–100 Dagstuhl Publishing Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany . 16 pages. https://strathprints.strath.ac.uk/45278/

- This is the companion article to the previous article, general video game playing
- This article mentions several other languages that have similar goals
- On page 4, the article elaborates on the criteria it considers necessary for VGDL to be a success
- On page 5, the article breaks down the aspects of the game the language must represent
- Starting at page 9, the article breaks down how space invaders, lunar lander, and frogger would be implemented in VGDL
- the article mentions that they have a working prototype built in Python that utilizes Pygame
    - The article ends with a statement that they will continue work on this prototype
- The article included the code for implementing space invaders using the prototype VGDL language

Although it seems like the goal of the language has shifted from describing the game to AI to quickly making testing environments for AI to interact with the language itself, it could still be a big help. For one, if the language was finished or at least significantly improved by this point, then there is the possibility that someone has already made a Tetris implementation of this language. Additionally, if Tetris has already been implemented in this language, it may very well have been used in the general game-playing competition; if so, the work of contestants in that competition could be very relevant to my own research.

Source 5: A Video Game Description Language for Model-based or Interactive Learning

- Tom Schaul. 2013. A video game description language for model-based or interactive learning. Artificial and Computational Intelligence in Games. Dagstuhl Follow-Ups, 6 . Dagstuhl Publishing, Wadern, pp. 85-100. ISBN 9783939897620. 8 pages. https://strathprints.strath.ac.uk/45278/

    - The author of this article is one of the authors from Towards a Video Game Description Language, Tom Schaul
    - This language was developed to generate diverse benchmark problems for learning and planning algorithms
    - This article contains much more comprehensive examples of VGDL syntax then the previous article, which was written when VGDL was much less developed
    - This article covers what the interface would look like for humans and, much more importantly, nonhuman players
    - The name of the language in this article is now PyVGDL, as it is essentially just a much higher-level version of PyGame written with the specifications of VGDL
    - Bots using this language interact with the game by taking one action per cycle
        - The bot has access to the getSensors and performAction methods

This language designed to be easily written and make games easy for AI to play could prove to be very helpful. This article states the PyVGDL supports continuous physics and object spawning, which is good if I plan to make or find an implementation of Tetris written in this language. A potential problem I have noticed is that no mention has been made yet of whether or not PyVGDL supports players or objects that take up more than one tile on the map.

Source 6: XML-Based Video Game Description Language

- Jorge R. Quiñones and Antonio J. Fernádez-Leiva. 2019. XML-based video game description language. Volume 8, 2020. Digital Object Identifier 10.1109/ACCESS.2019.2962969. 14 pages. https://ieeexplore.ieee.org/abstract/document/8945249/
    - This article is another video game description language based in XML instead of Pygame
    - This article mentions other VGDLs in section 2 and then ends the article by comparing itself to other VGDLs
        - This comparison includes a chart of which features are available in which VGDLs
        - The author believes that XVGDL is an improvement over the previously read about PyVGDL

- This language is set up to allow other engines and tools to be used
- When defining the player attributes, one can instead assign those attributes to a specified AI
    - This is set up to support externally implemented AI
    - AI can be assigned to NPCs as well
- All the work on this VGDL is available on GitHub
- The article ends with a full implementation of Pac-Man as an example of all the features previously mentioned being used.

I believe that with this more recent article, I have found a suitable environment for testing. The built-in support for importing other tools/extensions and the focus on AI support seem obvious advantages for my work. Going forward, I focus any reading on general game-playing AI themselves instead of the environment to test them in.

Source 7: An Evolutionary Approach to Tetris

- Niko Böhm, Gabriella Kókai, and Stefan Mandl. 2005. An Evolutionary Approach to Tetris. The Sixth Metaheuristics International Conference (MIC2005). 6 pages. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b0fe1ed14404db2eb1db6a777961440723d6e06f
    - This article states that it "describes the application of evolutionary algorithms to the popular game of Tetris."
    - The intro states that the computer chooses the best move by rating possible subsequent game boards as determined by a number of subratings
        - The weights of the subratings is determined by the genetic algorithm
    - The approach in this article goes with a heuristic approach to rating boards as the game does not allow unlimited time to place the tetriminos
    - On pages 2 and 3 the article lists 12 metrics by which a board is measured
        - Page 3 also includes the math functions used combine these metrics
        - This article acknowledges that the metrics used may differ from how humans assess the board
    - Ultimately the authors decided to use number of placed tetriminos to measure an AI's performance when playing Tetris, and thus the AI's fitness

    - Performance gains tended to drop off between generations 20 to 30 so that is were most tests were ended in the interest of time and getting onto the next set of parameters
    - The article concludes that the more specific the rating system used the less well the AI can be seamlessly moved to a new board size a keep the same performance

Source 8: Tetris Artificial Intelligence

- Wei-Tze Tsai, Chi-Hsien Yen, Wei-Chiu Ma, Tian-Li Yu. 2012. Tetris Artificial Intelligence. NTUEE, Taiwan. https://chihsienyen.github.io/pdf/Tetris_AI.pdf
    - This article attempts to get the AI to emulate human strategies such as clearing multiple rows at a time
    - This article names the specific Tetrominos,
        - I, J, L, O, S, T and Z.
        - Each is named after the letter in the alphabet it most resembles
    - The AI in this article choses not calculate every possible game state to save time
    - The equation to calculate a states value is shown on page 2
    - The tests in this paper were done in a tetris implementation using the game rules designed by Alexey Pajitnov
    - The team in this article tried three kinds of AI
        - ”Greedy A.I.” which always performs row elimination if possible.
        - ”Tetris A.I.” performs row elimination whenever the current Tetromino is ”I” and the height of block is larger than four.
        - ”Two Wide Combo A.I.” which focus not how many points it can get with a single action but on getting combo eliminations
    - This article concludes that ”Two Wide Combo A.I.” preformed the best over all

Source 9: Tetris tips from a seven-time world champion

- Jeff Ramos. 2019. Tetris tips from a seven-time world champion; How to begin mastering the classic puzzle game. Polygon. https://www.polygon.com/guides/2019/2/22/18225349/tetris-strategy-tips-how-to-jonas-neubauer

- This article is about an interview with Jonas Neubauer, a seven-time winner of the Classic Tetris World Championship.
- It's better to stack your blocks so that multiple pisces are cleared at once as this gives more points then the sum of that many individual line clears
- This article points out how you are able to increase the rate a piece falls at by using a soft or hard drop which respectively speeds up the fall and places the price instantly
    - Some implementations grant more points for faster line clearing and using hard or soft drops
- Clearing 4 lines at once is called a tetris and is done by making a 9x4 wall and then filling the last column with a I piece
- The interviewee, Jonas Neubauer, provides a few strategies, the most relevant of which are:
    - Playing flat: playing with the goal of keeping the top row as flat as possible so you are never short on options for where to place your pieces
        - Try not to have a peak or trough greater than 2
    - Build mounds in the center: if you have to make a mound higher then 2 make it in the middle
    - Understand the Rotation System: knowing the fewest amount of rotations to get a piece facing the right way will save time
    - Hold pieces to score: use storage to hold an I piece so it can be used to score a tetris
    - Delayed auto shift: hold down the direction you want to as soon as the last piece is placed, don't wait for the next one to appear
    - Look at the queue and the colors: learn to memorize the pieces shapes based on their color so you don’t need as much time to know what is coming next
    - Multiple spaces: try to leave a 2 wide space at all times so O’s, S’s, and Z’s always have a place to go