Principles of Pentris

Rafael Diederen, Aurelien Giuglaris Michael, Florentine Hegemann, Melis Kulbay, Sofia Ozhogina, Eden Rochman Sharabi
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands

Abstract

Table of Contents

Introduction

Tetris is a worldwide famous video game created by Soviet software engineer Alexey Pajiitnov in 1984. The rules of the game are simple: in the beginning, there is an empty grid. Then different Tetris pieces start to fall from the top, the player must then fit them the best way possible. They may rotate pieces and shift them to the left and to the right respectively. When one or more rows are filled, they are deleted, and all the rows shift down to meet the next incomplete row. The player gains score by clearing rows and loses when there is no space for new pieces to appear.

The classic version of Tetris has tetrominoes, elements composed of 4 squares. But what happens if we take bigger elements, for instance, pentominoes (elements containing five squares)? Thus, the primary aim of our research is to:

Build a user-friendly computer application to play Pentris on the grid of width 5 and height 15.

It is important to mention that our research was conducted with only educational purposes in mind, as many similar versions of Pentris have been produced and are widely available.

Creating agents' algorithms to play Pentris, we took as a basis our human intuition and experience in playing Tetris. The two main strategies we developed most are called LowestRowBot and SidesBot.

In terms of the structure of the report, the section "Methods" contains a more detailed explanation of the implementation of algorithms used in the code. Paragraph "Experiments" describes, analyzes and compares the data collected from experiments taken during the research. Section "Results" provides the reader with the outcome of the experiments. In “Discussion” the results are interpreted, and, finally, “Conclusion” summarizes the result of the whole study.

Methods

1. General overview

Our game/simulation consists of a playing field, an active movable piece (ActivePentomino), and one agent (human or Bot depending on the selection).

2. Active pentomino

The active pentomino is a pentomino that starts at the coordinates (0,0) on the playing field. It has a natural movement of y+1 for every 100 ms. The active Pentomino can be controlled by agents or by user input.

3. Agents

Our first 2 agents are model-based reflex agents with different condition-action rules. Our first model-based reflex agent, LowestRowBot, evaluates the previous state of the field (without the active pentomino in it) and based on the field it places the pentomino so that it is in the lowest row possible.

Our second model-based reflex agent, sidesBot, also evaluates the previous state of the field but places the pentomino so that it is in contact with as many sides as possible. This could be another pentomino, the side of the field or the bottom. To make this a viable agent we gave these different options a different score depending on their usefulness. We determined their usefulness by trial and error. We gave the sides 1 point, the pentominoes 2 points and the bottom 3 points.

After one of these agents determines the optimal position for the piece, we use a recursive function to calculate the path it needs to take, to reach this position. This prevents the agents from placing the piece in a position that isn’t reachable.

The whole path is calculated before the pentomino falls down for the first time. We give our players and agents 100 ms for this. Therefore, everything has to be calculated inside of the first 100 ms. Currently, all of the agents take less than 1 ms but this has to be taken into account in case new agents are made.

4. UI

The UI is an important part of the accessibility of our simulation. The top priorities are for it to be responsive and clear to understand. We are trying to do research, and do not need to sell our game. This means a simple UI with intuitive controls and fast response times is good enough for our purpose. We achieve this by using the standard java libraries and try to refrain from using any over-complicated UI functions. The general structure is divided into 4 parts: the buttons, the playing field, the High Score and current scores, and a field that listens to the keyboard for user input.

Experiments:

Experiments have been conducted by six users with six different devices with the following features:

Computer 1:

Processor: 2.3 GHz Dual-Core Intel core i5.

Memory: 8 GB of RAM 2133 Mhz LPDDR3.

Computer 2:

Processor: AMD Ryzen 3 3250U 2.6 GHz CPU

Memory: 8GB of RAM 2400 MHz

Computer 3 :

Processor: Intel(R) Core(TM) i7-8665U CPU 1.90GHz 2.11 GHz

Memory: 16 GB of RAM

Computer 4:

Processor: 11th Gen Intel(R) Core(TM) i5-1135G7 2.40GHz 2.42 GHz

Memory: 16 GB of RAM

Computer 5:

Processor: Intel(R) Core(TM) i3-7100U CPU

Memory: 8 GB of RAM 2.40GHz

Computer 6:

Processor: Intel Celeron CPU 1037U 1.80GHz

Execution of Game

The experiments in this section have been dedicated to provide users a bug-free, runnable game that contains the following features of a typical Tetris Game. These experiments are related to ensuring and maintaining the playability of our game. In the execution part, the experiments have been conducted mainly by running the programme and making the users to play the game to verify the functions of the implemented features.

Rotating, shifting or pushing the pentominoes left, right or down: This experiment has been done to ensure the playability of our programme. The method of experiment is to simply run the programme and play the game to evaluate whether the relevant methods of this feature are acting in accordance with the rotate, shift or push commands. The positions of the pentominoes have been verified on the UI following the commands.

Ignore Sides: While experimenting on the playability, a defect has been detected. When a pentominoe touches the left, right and or upper sides of the field, the programme has thrown an “out of bound” issue. Since this issue was directly related to the playability of the game, we conducted our experiment by simply running and making users play the game after the implementation of our relevant ignore sides method.

Score adding & Highest Score: These two features are one of the main features of the Tetris Game. Implementation and maintenance of these features in the execution part are highly related to the playability of our game. Therefore, the experiment that has been conducted in this section is again running the programme and playing the game in order to be able to verify the intended features.

<u>Bot</u>

After having made the game playable and implemented the “lowest row bot” and the “sides bot” playing the Tetris game with pentominoes, we conducted the following experiments to check the performance of the bots:

1 - Since our aim is to create an intelligent bot that would aim for passing the “highest score”, in our experiments’ the number of runs of the each bots and the relevant highest scores per each run received by the each bots have been registered.

Method: We ran the Lowest Row Bot and the Sides Bot the following times: 100, 500, 1.000, 5.000 and 100.000. We compared the number of runs and the highest score received per each bot

Research Question: Which bot has received the highest score per each number of runs?

	NUMBER OF RUNS	**100**	**500**	**1000**	**5000**	**100000**
BOT						
Lowest Row Bot	**HIGHEST SCORE**	14	19	18	21	34
Sides Bot		16	25	24	36	38

2 - The field of tetris is an important and relevant arena for AI researchers, but no matter how developed the agent is, it is still unable to beat the human master of the game (Algorta, & Şimşek, 2019). This is the reason why we described intelligence as “thinking humanly” and “acting humanly”. In order to evaluate these approaches, we conducted a Turing Test among the following audience. The aim of this experiment is to have a bot that has the most human-like use.

Sample: Team members and their relatives. Overall 20 subjects aged 18 - 70+.

Method: We recorded both of our bots playing realistic games and one of our team members. We showed all 3 videos to each subjects. At the end of every video each subject had to determine if it was a bot or a human playing.

Research Question: Was the game played by a human or a bot?

3 - The last experiment we conducted is to measure the performance of the bot versus the real users. As stated above, AI could have never beaten the human master of the game. Therefore, we wanted to compare the highest score received by a human and a bot in ten different occasions.

User Experience

GUI plays a major role in measuring the user experience. If the interface is not perceived as "user-friendly" or simply easy to use, this would affect the user experience negatively in a way to discourage users from using the programme (Low Tze Hui, S., & See, S. L, 2015).

One way of measuring the UI experience is to conduct a survey among the target audience of our game by making them actually use the programme, namely play our Tetris Game.

In our survey we asked for age and level of degree to make sure that the individuals fit in our sample.

We also asked for their level of gaming generally, as well as specifically experience with Tetris. This is to see whether there is a difference in how

intuitively the game is perceived depending on how much experience the individual has. Optimally the game should be as intuitive for someone that has never played, as for someone that is a professional gamer.

Sample: Members of team 11 and their family. Overall consisting of about 30 subjects each aged between 18 and 70+.

Method: We prepared a survey (see appendix 1) and made the survey to the sample.

Efficiency of Algorithm

In order to evaluate the efficiency of the algorithm, we measured the running time of the programme and the memory usage per each of the six computers with features stated at the beginning of the "Experiments" section.

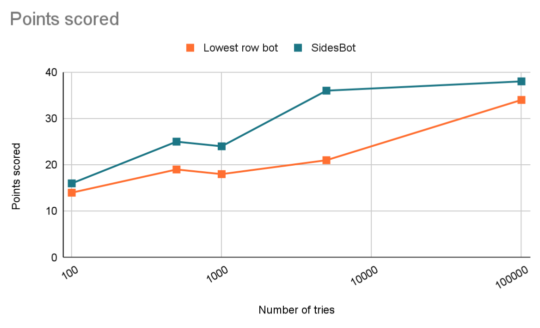
Results

Agent Sections Results :

Having 2 bots, an experiment was conducted to test and compare their efficiency.

Answering the following research question: Which bot has received the highest score per each number of runs?

Results are the following:



The Sides bot curve is higher than the Lowest Row bot curve for every run/simulation from a 100 simulations to 100 000.

To add further information on our experiment :

BOT	NUMBER OF RUNS	100	500	1000	5000	100000

Lowest Row Bot	**HIGHEST SCORE**	14	19	18	21	34
Sides Bot		16	25	24	36	38

Complexity Time for Bots (time it takes to run program) Results:

Sides bot is more efficient than lowest row bot , meaning it achieves a higher score on every run/simulation , but is it the quickest : Which one of the them is the quickest to decide which pieces to play? This is how we could test complexity. (2) using // time module? (start - end).

Sides Bot : // Not provided
Lowest Row Bot: // Not provided

Runtime CPU results :

Run the programme on 6 of the computers and evaluate CPU taken for running it.
Computer 1 takes 21%CPU and 300 MB of RAM
Computer 2 takes 25% CPU and 480 MB of RAM
Computer 3 takes 17% CPU and 400 MB of RAM
Computer 4 takes 19% CPU and 500 MB of RAM
Computer 5 takes 30% CPU and 650 MB of RAM
Computer 6 takes 27% CPU and 560 MB of RAM

GUI Section Results

The game has been played by members of the team considered as young and by elder people (active in society already) mostly members of our respective families (siblings - Parents - GrandParents).

Each of them gave us a feedback on the runtime they thought was most appropriate for playing the game.

Age	Situation	Time Feedback
18	Student	100ms
18	Student	100ms
18	Student	100ms
23	Student	100ms
23	Student	100ms
29	Student	100ms
59	Working	Between 100ms and 1000ms 100ms : too fast 1000ms : definitely too slow
37	Working	100ms
54	Retired	1000ms
62	Working	1000ms
71	Retired	Between 100ms and 1000 ms 100ms : too fast 1000ms : too slow
17	Student	100ms
20	Student	100ms
21	Working	100ms
22	Student	100ms
25	Student	100ms

27	Working	100ms
81	Retired	1000ms
89	Retired	Between 100ms and 1000ms 100ms : too fast 1000ms : too slow
51	Retired	1000ms
56	Working	1000ms
16	Students	100ms
25	Students	100ms
56	Working	Between 1000ms - 100ms : 1000ms : too slow 100 ms : too fast
56	Working	Between 1000ms - 100ms : 1000ms : too slow 100 ms : too fast
65	Retired	100ms
53	Working	1000ms

Younger people between 18 - 37 prefer 100ms for playing the game

Elder people whether they are active in society or retired prefer either : 1000ms or between 100ms and 1000ms.

The experiment here was used to determine how players experiences playing our game.

It is a way to evaluate our GUI and see if it is appealing enough for players and to answer our main question : Creating a playable Tetris game with BOT and GUI.

Results performed on the survey in appendix 1. // not published yet

Turing Test Results :

(Summary experiment : Can people differentiate bot player from user player).

Subjects :	Video 1 Human (Answer)	Video 2 : Lowest Row Bot (Answer)	Video 3 : Sides Bot (Answer)
Team member : Melis 29Years	human	bot	bot
Team member : Flo 23Years	human	bot	bot
Team member : Rafael 18Years	human	bot	bot
Team member : Sonya 18Years	human	bot	bot
Team member :Aurelie	human	bot	bot

n 18Years			
Team member : Eden 23Years	human	bot	bot
Male : 60Years	human	bot	human
Female 54 Years	bot	bot	human
Male : 62 Years	bot	human	bot
Female : 37 Years	human	bot	bot
Female : 71 Years	bot	human	human
Female : 21 Years	human	bot	bot
Female : 16 Years	bot	human	bot
Male : 45 Years	human	bot	bot
Male : 21 Years	bot	human	bot
Female : 53 Years	bot	human	human
Male : 60 Years	human	bot	bot
Female : 25 Years	human	bot	bot
Female : 27 Years	human	bot	human

All team members got the test right , having seen the agents many times. The other subjects (relatives from team members) had mixed answers.

Discussion

The field of agents to solve problems has been rising for the last thirty years. “Weak” AI (Chowdhury & Sadek, 2012), just like the agents that were developed, are narrow to a specific field or task. More complex systems would still be weak AI, for instance, state of the art agents such as multi-agent reinforcement learning playing StarCraft II (Vinyals, Babuschkin, Czarnecki, Mathieu, Dudzik, Chung, & Silver, 2019). This mix of algorithms combines several techniques for neural network architectures, imitation learning, reinforcement learning, and multi-agent learning (2019). No matter how advanced this AI is, it only fixes a Starcraft 2 game.

The field of tetris is an important and relevant area for AI researchers, but no matter how developed the agent is, it is still unable to beat the human master of the game (Algorta, & Şimşek, 2019).

Some early attempts suchs as Lagoudakis, Parr, & Littman measure the mean column height and the sum of the differences in consecutive column heights (2002). In addition, by using the least-squares policy iteration, the agent had an average score of 1000 and 3000 lines (Lagoudakis, Parr, & Littman, 2002).

Another milestone was accomplished in 2006 by the researchers Farias and Van Roy where they used Bellman equations to solve tetris; this approach solves 4500 lines (2006).

Issues we had, how they could have been solved: Agents calculated positions that were impossible to reach -> solved through tracing full path before falling down (recursion);

Even though our results show that the game is relatively successful when tested by samples of college students, those samples are rather small. Therefore, the power of these tests might not be sufficient to draw any proper conclusions.

Another limiting factor for this research was time pressure. This project had to be done in about seven weeks, leaving us with enough time to come up with the different agents and a playable game, but not necessarily allowing for us to fully develop the game in a way for it to be distributed. Some of the details for the game were also pre-given, such as the size of the field and the different Pentominoes. The field size especially makes for a harder to play game. Here a narrower field size would have been a better option for an easier playable game.

starcraFT chess openAI bostonDynamics

Conclusion

Our main research Question was if we could set up a user-friendly computer application of Pentris in a grid of (5, 15). This was moderately successful, as the interface was found to be intuitive and user-friendly. However, the game was perceived as too demanding and not enjoyable. If someone was to replicate our game in a way that makes it more enjoyable, they should definitely opt for a field that is broader. This way none of the pentominoes would result in a row being deleted all by itself. For the optimal playability the different sizes of the game should be empirically tested.

Moreover, more complex agents could be developed, for instance, agents based in genetic algorithms, neuronal networks or simulated annealing would be interesting to implement for further research. In comparing these more complicated agents to the ones developed in this research, one could show how much better these agents are in terms of getting to a higher score or passing the turing test.

This could in the end show whether there is always a necessity for more complex agents or whether this is even useful, given that these are also harder to implement.

To summarize, the first agent is a solid viewpoint to stand up and reach the next step of human evolution.

References

Chowdhury, M., & Sadek, A. W. (2012). Advantages and limitations of artificial intelligence. *Artificial intelligence applications to critical transportation issues*, *6*(3), 360-375.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., ... & Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350-354.

Algorta, S., & Şimşek, Ö. (2019). The game of Tetris in machine learning. *arXiv preprint arXiv:1905.01652*.

Lagoudakis, M. G., Parr, R., & Littman, M. L. (2002). Least-squares methods in reinforcement learning for control. In *Hellenic conference on artificial intelligence* (pp. 249-260). Springer, Berlin, Heidelberg.

Low Tze Hui, S., & See, S. L. (n.d.). 6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, AHFE 2015. In *Enhancing user experience through customisation of UI design*. Retrieved November 30, 2021, from www.sciencedirect.com.

APPENDIX 1

Pentris Game

How Are We Doing?

In this survey, we will ask you a few Questions about the playability of our Pentris Game. For this you will have to answer as honest as possible. Try to answer these Questions right after playing our game.

Do you agree to take part in our survey and can confirm that you are over 16 years of age?

☐ Yes ☐ No

What is your age?

What is the highest degree you have completed?

☐ No schooling completed ☐ High school graduate ☐ Associate Degree
☐ Bachelor's Degree ☐ Master's Degree ☐ Doctorate Degree

How would you describe your experience with gaming generally?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Completely inexperienced (Never played) — Very experienced

How would you describe your experience with Tetris specifically?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Completely inexperienced (Never played) — Very experienced

How would you rate your experience playing our Pentris Game?

[✓ 1] ☐ 2 ☐ 3 ☐ 4 ☐ 5

Disappointing — Exceptional

How intuitive was the Gameplay?

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Hard to understand — Very intuitive