DEPARTMENT OF COMPUTER COMPUTER SCIENCE
ACADEMIC YEAR 2022/2023

November 2022

PROJECT PROPOSAL

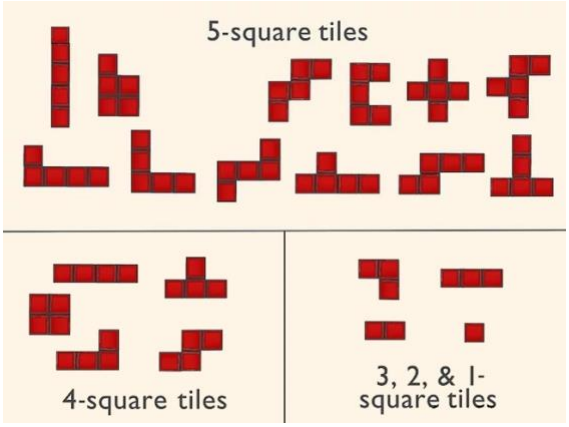
BLOKUS GAME

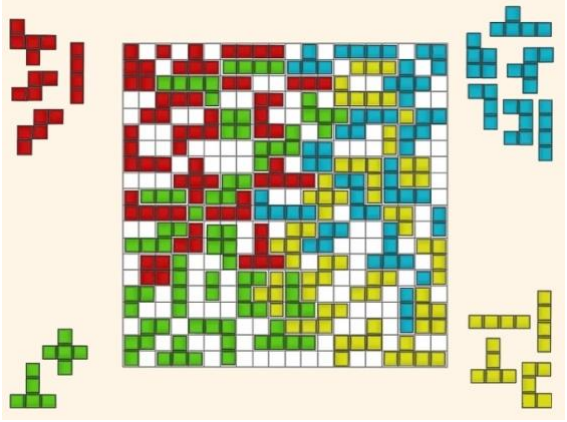
DESCRIPTION

The Blokus game is a deterministic game for two and four players (that is Blokus Duo and Blokus Classic). Blokus is an abstract strategy board game, invented by Bernard Tavitian and first released in 2000 by Sekkoïa, a French company. Throughout the project, I will create a playable Blokus game from a scratch using Pygame, implement basic and more advanced strategies for a computer player, and test, evaluate, and upgrade them. To achieve a strong strategy, I will gather the information from a game and combine them. I explain it in more detail in the "Aims & Objectives", and "Development & Implementation Summary" sections. To obtain the needed knowledge, I researched previous projects about the Blokus Game and understood the algorithms developed in them. Researched materials are explained in depth in the "Key Literature & Background Reading" component. To properly understand all the above, I will now introduce the rules of Blokus Classic and Blokus Duo [*Figure 1*]. A detailed explanation can be found on Wikipedia's website [1].

Rules for Blokus Classic:

"The game is played on a square board divided into 20 rows and 20 columns, for a total of 400 squares. There are a total of 84 game tiles, organized into 21 shapes [illustration below] in each of four colours: blue, yellow, red, and green. The 21 shapes are based on free polyominoes of from one to five squares (one monomino, one domino, two triominoes, five tetrominoes, and 12 pentominoes). The order of play is based on colour, with blue going first, followed by yellow, red, and green. The first piece played of each colour is placed in one of the board's four corners. Each new piece played must be placed so that it touches at least one piece of the same colour, with only corner-to-corner contact allowed - edges cannot touch. When a player cannot place a piece, he or she passes, and play continues as normal. The game ends when no one can place a piece. When a game ends, the score is based on the number of squares in each player's unplayed pieces; a player loses one point for each square (e.g., a tetromino is worth -4 points). If a player played all his or her pieces, he or she gets a bonus score of +20 points if the last piece played was a monomino, +15 points otherwise." The end state of a Blokus game is presented in an illustration below.





Rules for Blokus Duo:

"Blokus Duo is for two players only and uses a smaller (14×14) board; the pieces are of purple and orange colours. The two starting squares are placed, not in the corner (as in the original Blokus game), but nearer to the centre ((5,5) and (10,10)). This makes a crucial difference in the flavour of the game because players' pieces may (and usually do) touch after the first move. Even more than with the original game, Blokus Duo is an offence-centred game; it is also a much purer strategy game than the four-player game since one is not in danger of getting ganged up on by three other players (as sometimes happens with the four-player version)."

Figure 1. The rules of the Blokus Game

Key Literature & Background Reading

To appropriately prepare for the project, set goals and don't make common mistakes, I started with background reading. Unfortunately, I haven't managed to find books that would either introduce simple 2D game engines or describe algorithms that would be worth considering at this point of a project. On the other hand, using resources available online, I found intriguing strategies that will be vital in the development of algorithms based on them. Since Blokus Duo is easier solvable, playing against only one opponent, most of the strategies were developed with the intention to work in Blokus Duo; Thus, they may not be successful in Blokus Classic. In the following lines, I will introduce the information obtained from mentioned resources and describe how I am going to use them in this project.

Blokus Classic:

Figure 2. Barasona opening.

It may be worth considering starting with the Barasona opening [Figure 2]. The basic idea is that your first four pieces leave no possibility for an opponent to move "through" your corner, without the use of his one-piece. Although it's a very efficient opening, it may be more useful in one strategy and less in others as the consequence of "blocking" other players is that we will reach the centre of the board after placing 4 pieces instead of 3. It moves the centre of the board in our direction.

While playing Blokus Classic, it pays to not be confrontational. If the board splits into two 1v1 games, and players A and B are being confrontational and denying each other access wherever possible, and players C and D are being semi-cooperative and leaving gaps to work around each other, then players C and D will naturally get more of their pieces onto the board. If there's a region you've trapped all the other players out of, then you'll only be able to take advantage of about half the space. Whereas a region twice the size, shared between two players, can let both those players get twice as many of their pieces down if they leave space for them. As a result, it's usually best to place a shape in a region where there is one opponent at a given moment.

Blokus Duo:

As mentioned in the Blokus Game Solver [2], there is an evident first-turn advantage that will need to be considered during testing and evaluation. Furthermore, the author implemented two greedy algorithms (depending on a size of a piece and the difference in available corners for you and your opponent after playing it) as well as a random strategy. It appeared that the greedy (basic) algorithm is stronger than the greedy (advanced) algorithm (i.e., randomly choosing one of the largest available pieces is more effective than calculating their weight using the number of available corners and the size of a piece). I believe that the weight calculation was not detailed enough to be considered correct (2*size + corner_difference); however, it may be worth noticing that a calculation not detailed enough may be disadvantageous for a strategy, and that there exists a stronger algorithm created from using the same data.

Furthermore, the project worth noticing is the Blokus Duo game on FPGA [3] where basic strategies were combined and modified throughout the game resulting in a strong strategy overall [*Figure 3*]. This proves that with appropriate testing and evaluation it is possible to achieve an effective dynamic strategy using simple data such as the number of possible places where an opponent can place a piece, the number of possible places where we can place a piece, using the largest piece first, and by taking strategic places (i.e., points that let us escape from the other player's trap).

TABLE I. WHEIGHT OF EACH STERATEGY DURING THE GAME

Phases	Strategies			
	Disrupt other player	***Increase chance of your next move***	***Bigger tiles***	***Strategic points***
Start of the game	High	High	Very High	Normal
Middle of the game	High	Normal	High	Very High
End of the game	Low	Low	Normal	Very High

Figure 3. The dynamic combination of simple strategies

Projects on the Blokus Game that used more advanced strategies [4] [5] used a min-max algorithm. It will only be efficient to use this algorithm in the Blokus Duo as it is not possible to count all the possible variations when playing against 3 opponents in the Blokus Classic. The information that may be useful in the Blokus classic is what values mentioned projects used to assign weights to moves. The first of them [4] chooses 4 top candidate moves. Then, for each of the four selected candidates, 4 top moves of the opponent, a total of 16 candidates, and so on with depth 4. It may be useful to use this strategy in the middle/end stage of the game. The second of them [5], uses piece squares (squares occupied by a piece), candidate squares (empty squares where we can place a shape – they touch a corner of one of our squares) and area of influence (a certain neighbourhood area of candidate square) [*Figure 4*]. The next move maximises the area of influence of the self-player and minimises that of the opponent.

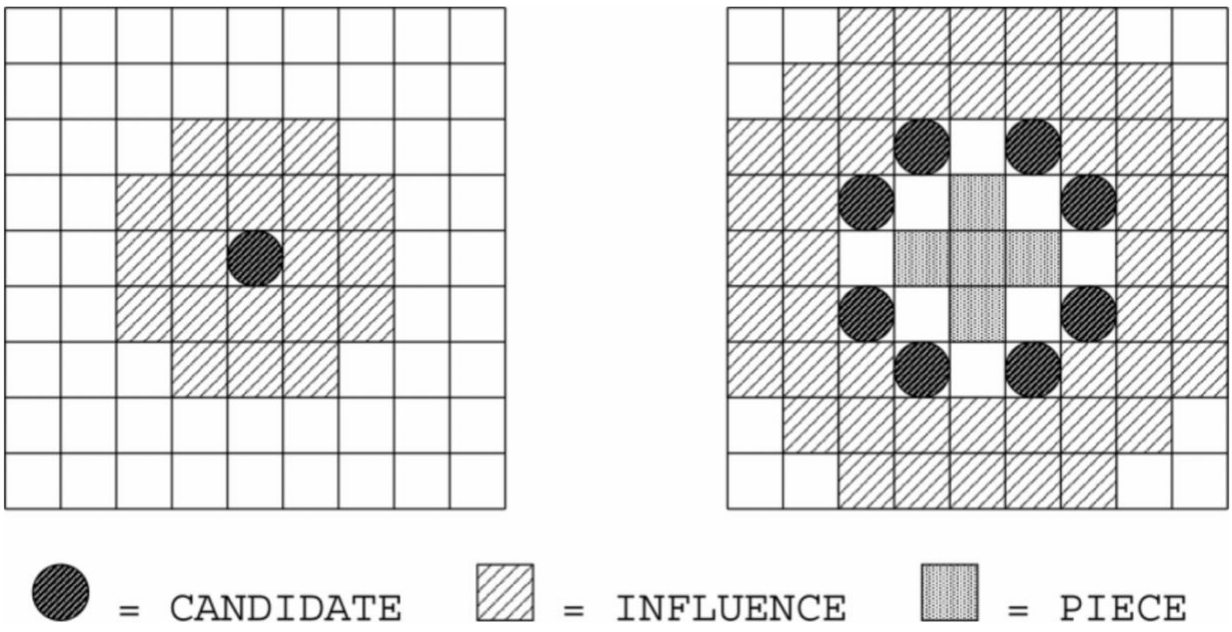


Figure 4. The illustration of choosing candidate, influence, and piece squares.

Concluding the information above, it is possible to develop a strong mathematical-based algorithm for Blokus Duo, possibly using the Barasona opening to significantly reduce the number of computations because of reducing the depth. In the case of the Blokus Classic, it is crucial to dynamically combine different strategies, depending on the time of the game, whilst remembering that too compendious calculation may be more destructive than beneficial. Based on the information above, I will formulate the Aims & Objectives for the Blokus Game project.

Aims & Objectives

Bullet points below identify the Aims, followed by the general objectives necessary to achieve them.

- Create a playable Blokus game.

  Objectives:
  1. Create a board and pieces in Pygame.
  2. Set the rules of the game.
  3. Terminate the game at the right time.
  4. Programming mechanism of moving, rotating and placing a shape.

- Implement mechanics of placing a shape

  Objectives:
  1. Indicate if it's possible to place somewhere a shape or not.
  2. Pass a turn if it's not possible to place a shape or finish the game.
  3. Pick a piece, move it, rotate it, and put it in a specific place.
  4. Update the set of available pieces.

- Implement basic computer player strategies

  Objectives:
  1. Implement a greedy algorithm – the largest piece first
  2. Implement a greedy algorithm – Increase chances of your next move
  3. Implement a strategy - block your opponent
  4. Implement a strategy - Choose an area where there are fewer players but more than 0.
  5. Implement a strategy - Maximise Influence pieces for you and minimise for the opponent.

- Implement a strong computer player strategy

  Objectives:
  1. Test basic computer player strategies.
  2. Find out which basic strategies are stronger in a particular state of the game.
  3. Combine basic computer player strategies.

- Implement a mathematical-based algorithm for Blokus Duo

  Objectives:
  1. Count the number of candidate squares for you and your opponent before and after a move
  2. Count the number of possible different shapes to place on the board.
  3. Check the current difference in points.
  4. Check if there are any squares that the opponent won't have access to in the future - are "reserved" for our piece.
  5. Count the influence area for both players before and after the move.
  6. Find the most optimal correlation between obtained data.

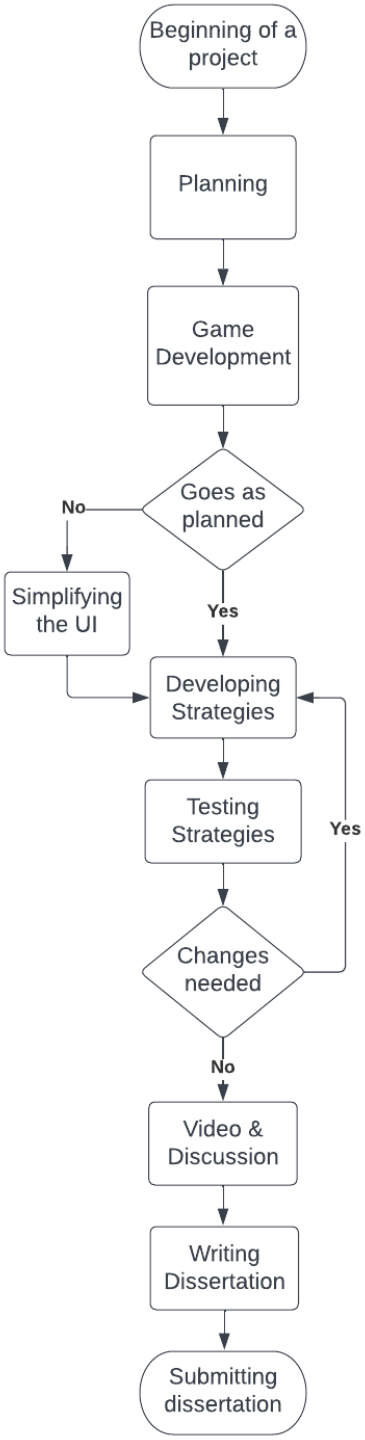


Figure 5. The Workflow Diagram

Development & Implementation Summary

The most important part of the project is choosing the proper development environment followed by implementing the constraints of the game and strategies.

Blokus game is an easy game that doesn't require anything else, but a 2D board and 2D pieces. That's why I decided to develop it using Pygame. Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. With Pygame I can use my programming skills, as well as have a graphical representation of the game. I have never used python; However, Pygame offers everything that my Blokus game needs; Thus, I decided to choose it over other frameworks.

The board of the game can be represented as a 20x20 array of 0s. Pieces can be represented as a 5x5 array of 1s and 0s (2s and 0s for the next player and so on). Once the piece is allowed to be placed somewhere, the piece's array will be added to the proper part of the board array. This way it will be possible to define empty squares and squares occupied by each of the players (0, 1, 2, 3 or 4).

To develop algorithms and strategies, I will gather information mentioned in the Aims & Objectives, based on the current state of the board, and I will calculate them for both players, before and after the considered move. For most of the data, it is only needed to check some parts of the board (occupied by 1s 2s, 3s or 4s) and some areas around it. Thanks to it, it is possible to significantly reduce the number of checks and combinations. The vital part of developing every algorithm will be testing and evaluating the obtained results.

A workflow Diagram presenting the process of the project is included on the left. In the next paragraphs, I will describe how I am going to test developed strategies and show how my project meets BCS project criteria.

Data Sources

I am not using any public data; all the code will be written by me.

Testing & Evaluation

I will test algorithms in Blokus Duo, as well as in Blokus classic against other people. No data but the final score will be gathered (win, draw, lose). I will test computer players against other computer players that use different strategies. I will use the result and every state of the game to evaluate the strategies and improve them.

Ethical Considerations

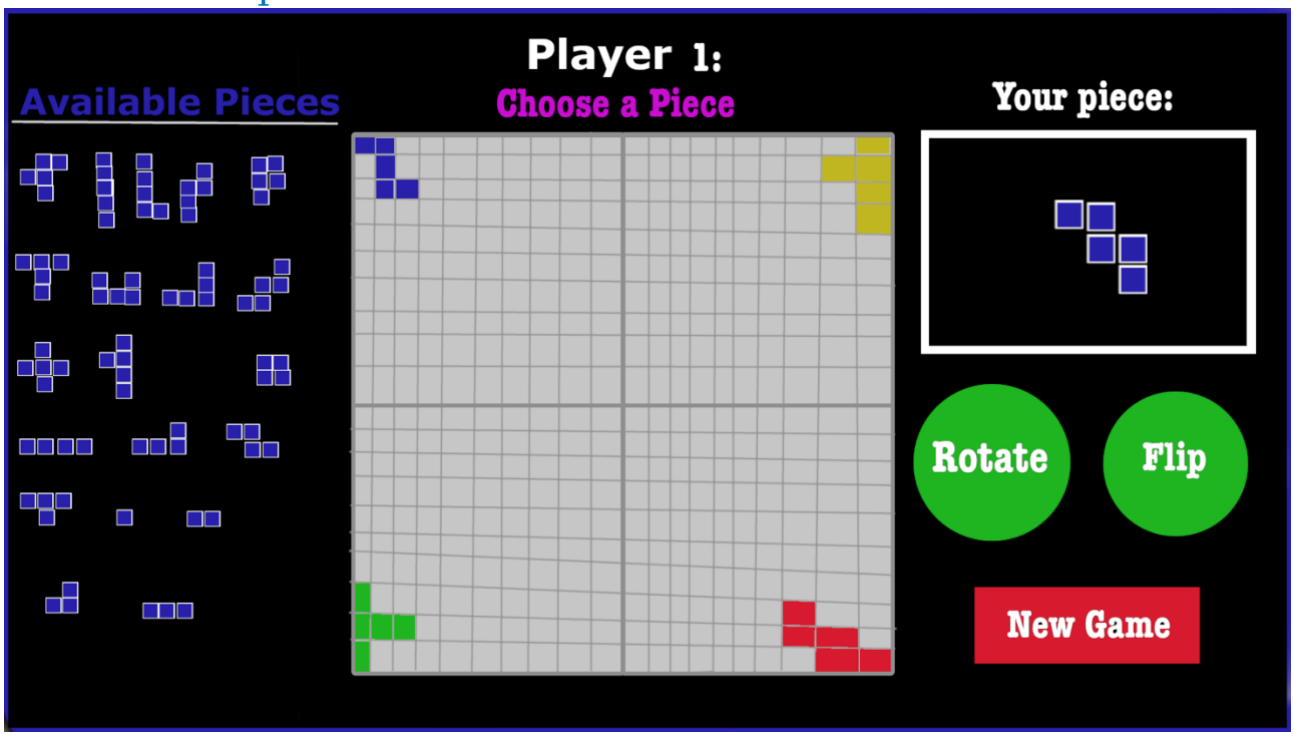
I confirm to have read the university's ethical guidelines (Ethical Conduct, 2022) and I will follow them.

BCS Project Criteria

- The project involves python programming skills and the wide knowledge of algorithms learnt during the degree programme. Analytical skills are needed to develop a strong strategy for a computer player.
- Creativity will be needed to create a friendly and intuitive user interface.
- The project combines programming skills, and algorithms knowledge, as well as the evaluation and creation of a quality strategy consisting of them.
- Thanks to finding a quality solution for a computer player, it will be possible to derive a great strategy for a human player that can be used while playing the Blokus game.
- The whole game will be developed from scratch. It will be possible due to the simplicity of the Python language and Pygame computer graphics implementation.
- I have strong programming skills and mathematical background; Thus, I believe that I'll be able to follow the development steps and fully deliver the project by the deadlines. It's important to mention that I've never used python before; However, knowing many other programming languages, I shouldn't have any major problems along the way.

UI/UX Mockup

Figure 6. UI/UX Mockup



The user interface, presented above, will work in the following way: In the beginning, users will be able to choose if they want to play Blokus Duo or Blokus Classic, and choose the number of computer players and their strategies. In the middle, above the board, a message about the current action required will be displayed. During the game, the user will be able to pick a shape from the left side "Available Pieces". The chosen piece will appear on the right side in the window "Your piece:". That's where the user will be able to flip and rotate it. Once the piece is rotated the way the user wants it to be, they will be able to place it on a board by clicking

on it. Once the piece is used, it will be removed from the section “Available pieces”. The next user’s turn will start. Pieces on the left will change colour.

To properly conduct the project, I created a project plan that includes scheduled actions that I will undertake during the upcoming months. Subsequently, I will describe possible risks that may occur during the project and contingency plan to accurately prepare for them and quickly react if they happen.

Project Plan

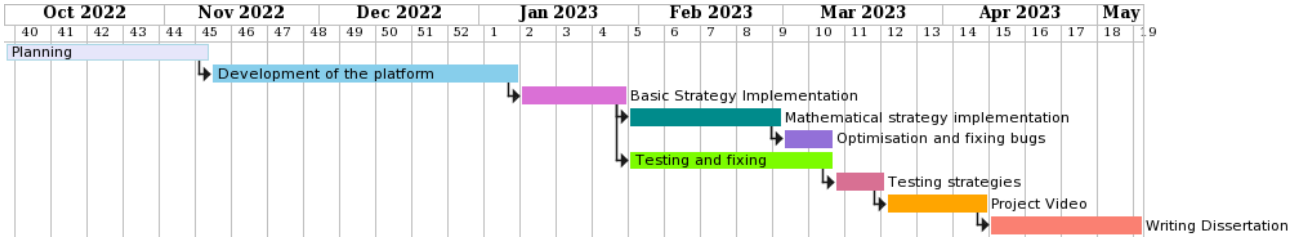


Figure 7. Project Plan

Risk	Contingency	Likelihood	Impact
Programming problems	Organised programming since the very beginning with comments. Learning from available resources on GitHub and YouTube along the way.	Medium	Very high - it won’t be possible to fully develop the project.
Running out of time	Detailed project plan at the beginning and consistency in development.	Medium	Low - Some part of the project wouldn’t be developed as much as wanted but it wouldn’t impact other parts.
Software Failure	Testing software and fixing errors along the way.	High	Very Low - They will be fixed right away. Errors are part of programming.
Losing all the files	Backup of code on a physical drive.	Low	Very High - it would cause starting from the very beginning.
UI development problems	Creating playable game (easier programmable) but with worse UX.	Low	Low - it wouldn’t influence algorithms.

Risks & Contingency Plan

Figure 8. Risks & Contingency Plan

References

[1] Wikipedia. (2020). *Blokus*. [online] Available at: https://en.wikipedia.org/wiki/Blokus (Accessed: 1

November 2022).
[2] Chao, C. (2018). *Blokus Game Solver Program Blokus Game Solver*. [online] Available at: https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1305&context=cpesp (Accessed: 1 November 2022).
[3] Jahanshahi, A., Taram, M.K. and Eskandari, N. (2013). *Blokus Duo game on FPGA*. [online] https://ieeexplore.ieee.org/abstract/document/6714256 (Accessed: 1 November 2022).
[4] Borhanifar, H. and Zolnouri, S.P. (2014). *Optimize MinMax algorithm to solve Blokus Duo game by HDL*. [online] https://ieeexplore.ieee.org/abstract/document/7082821 (Accessed: 1 November 2022).
[5] Sugimoto, N., Miyajima, T., Kuhara, T., Katuta, Y., Mitsuichi, T. and Amano, H. (2013). *Artificial intelligence of Blokus Duo on FPGA using Cyber Work Bench*. [online] https://ieeexplore.ieee.org/document/6718427 (Accessed: 1 November 2022).