University of Bucharest
Faculty of Mathematics and Computer Science
Department of Computer Science

Alexandra Diaconu
Bogdan Alexe
MSC in Computer Science
April 2025

Computer Vision - Project 1

Qwirkle score calculator

Objective

The goal of this project is to develop an automatic system to calculate the score for the game of Qwirkle and its extension Qwirkle Connect.

Qwirkle Classic and Qwirkle Connect

Qwirkle is a strategic, family-friendly tile-laying game that challenges players to match pieces based on color and shape. The game pieces (often called "tiles") come in six different shapes and six different colors. Each shape-color combination appears exactly three times in the set, creating a total of 108 tiles. *Qwirkle* in its classic form is played directly on a flat surface (Figure 1 left), typically the tabletop itself. Players place tiles adjacently to form lines of matching shapes or colors, with no physical boundary or predefined layout. The objective is to score as many points as possible by placing tiles in strategic positions and completing *Qwirkles* - lines of six tiles that form a complete set. The game can be played by two to four players.

Qwirkle Connect is an extension that builds upon the classic Qwirkle rules. *Qwirkle Connect* employs a dedicated *board* to guide gameplay (Figure 1 right). The board is typically subdivided into squares (often configured in four interchangeable quadrants), with some

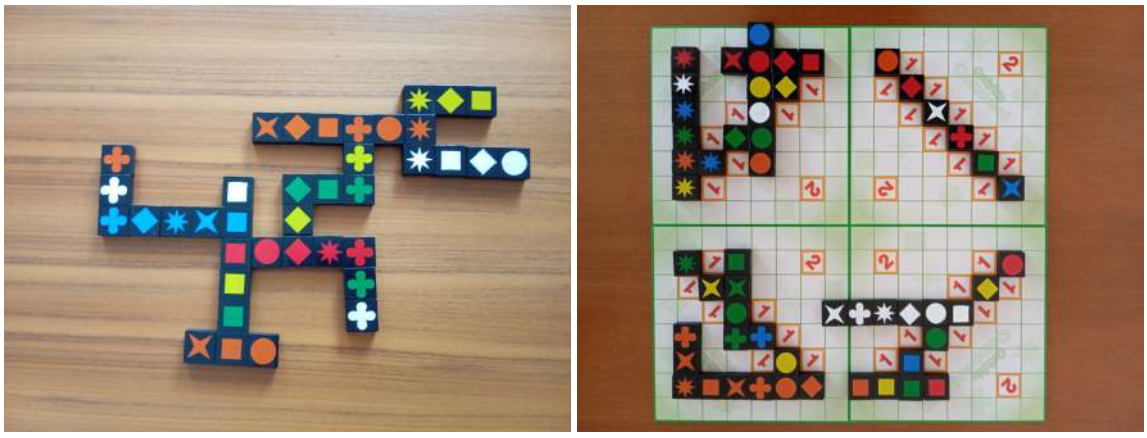


Figure 1: *Qwirkle Classic (left) vs Qwirkle Connect (right) gameplay.*

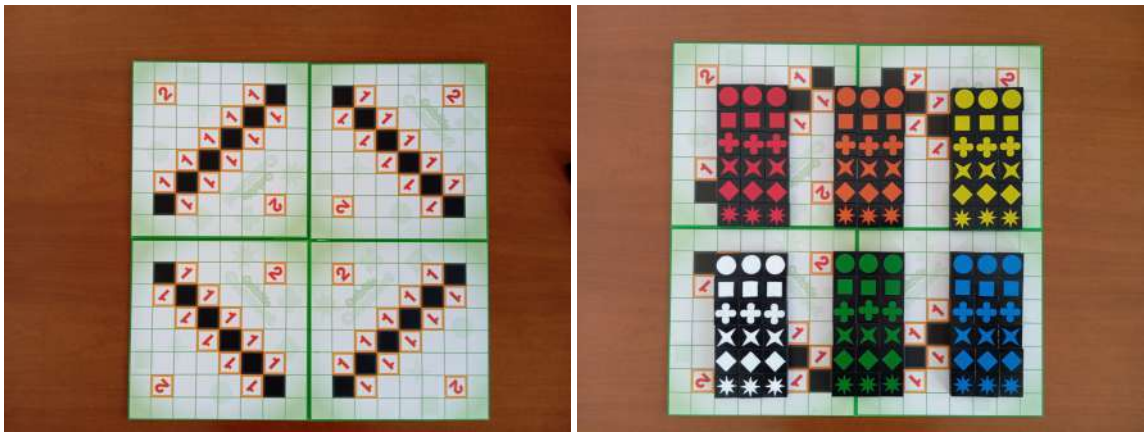


Figure 2: *The Qwirkle Connect board (left) and all possible tiles placed on the board (right).*

special squares offering *bonus points* or anchors to place initial tiles. As in the original Qwirkle, players use 108 tiles combining six shapes and six colors, forming lines that match either shape or color, without repeating tiles.

In this project, the primary focus is on the *Qwirkle Connect* variant, which employs a dedicated board (Figure 2) for gameplay. In contrast, the bonus section addresses *Qwirkle Classic*, where play occurs directly on the table without a board.

Tiles

In creating lines on the board, players use *tiles* that combine a shape with a color. There are in total 36 different types of tiles, each corresponding to a unique combination of shape and color. Each combination appears three times in the game, resulting in a total of 108 tiles. The tiles are composed as follows:

- six distinct shapes: circle, square, diamond, four-point star, clover, and eight-point star;
- six distinct colors: red, orange, yellow, green, blue, and white;
- each shape appears in each color exactly once, for a total of $6 \times 6 = 36$ unique tiles;
- each unique tile appears three times, resulting in $36 \times 3 = 108$ tiles in the full set.

Board

In *Qwirkle Connect*, players use a modular board divided into four identical quadrants. Each quadrant contains the same arrangement of *special squares* that can be used in the initial phase of the game or otherwise enhance the points scored from placed tiles. This modular design allows players to *reconfigure* the board before each game by rotating or swapping these quadrants, producing a variety of layouts (Figure 3). When all four quadrants are placed together (in whichever configuration players choose), they form a seamless

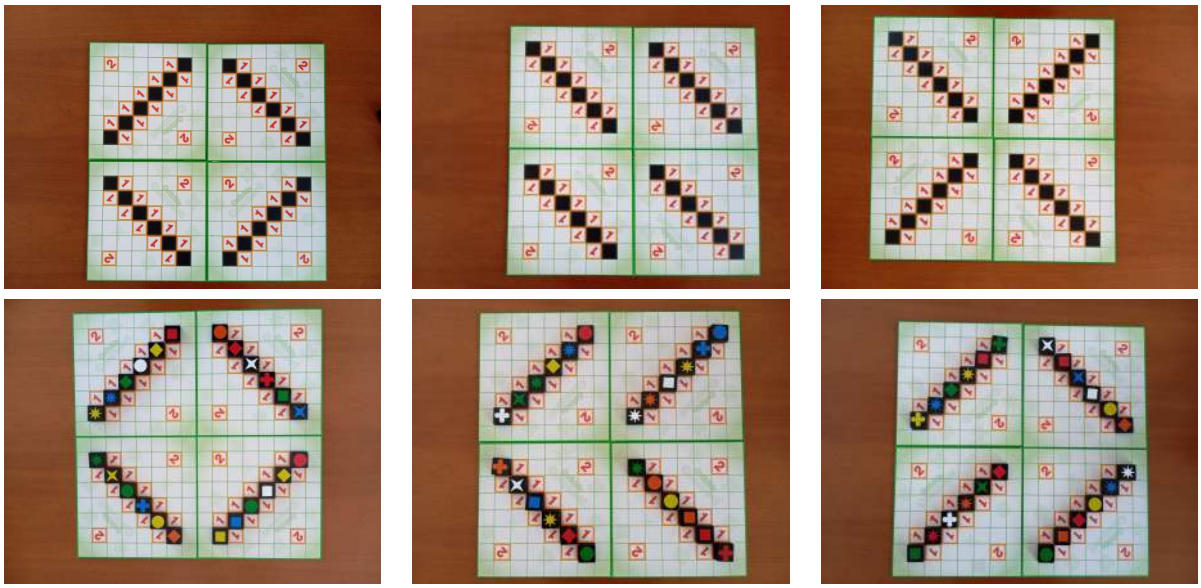


Figure 3: *Different layouts of the board*

playing surface. A number of squares within each quadrant are specially marked. These include:

- *six black squares* - these are used at the beginning of the game to place six initial tiles there;
- *additive bonuses squares* - these are used to award +1 or +2 bonus points whenever a player places tiles on these squares. There are ten squares with +1 bonus point and two squares with +2 bonus points.

Since each quadrant is identical, these special squares appear in the same relative positions, but may end up in different locations on the final board depending on how the players rotate or swap the quadrants.

Basic placement rules

A player's turn involves placing one or more tiles in a single row or column such that several rules are fulfilled:

- **same color or same shape, no repetitions.** All placed tiles must share a single feature (color or shape), and the current player cannot include duplicate shape-color combinations within the same line.
- **connectivity requirement.** Each newly placed tile (or group of tiles) must connect to at least one existing tile on the board.
- **one continuous line.** The current player may place multiple tiles during a turn, but they must all lie in a single row or column, forming a straight line with no gaps.

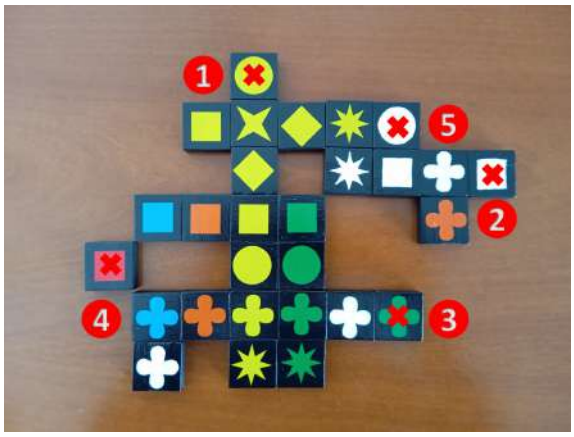


Figure 4: Example of some valid and invalid placements.

Valid and invalid placements of tiles

Figure 4 illustrates several valid and invalid placements. Below, we detail the causes of each invalid placement :

- **invalid placement 1** (labeled "1" in Figure 4): this placement contains seven yellow tiles in one line, exceeding the maximum allowed of six. The issue arises because the newly placed yellow circle duplicates a shape/color combination already present in that line.
- **invalid placements 2 and 3** (labeled "2" and "3"): each placement reuses a tile that was already part of the line (specifically, the white square and the green clover), violating the rule that a shape/color combination may appear only once per line.
- **invalid placement 4** (labeled "4"): the red square tile is placed in a location that does not connect to any existing line, thus breaking the connectivity requirement.
- **invalid placement 5** (labeled "5"): the white circle tile does not match the yellow tiles in the line it touches, so color-based matching is not satisfied.

The play

We consider the scenario with only two players, Player 1 and Player 2. At the beginning of the game, Player 1 arranges the four quadrants in any orientation he likes, ensuring they form a contiguous playing surface. He also then places twenty-four tiles randomly drawn from the bag on all black squares, six in each quadrant (Figure 3 row 2). This initial placement becomes the *central structure* that subsequent moves must connect to.

After this initial step, each player receives six tiles drawn from the bag with all remaining tiles (initially there are 108 tiles in the bag, but 24 of them are placed on the black squares in the initial stage so there are only 84 tiles left in the bag). At each turn, the current player places one or more tiles from their hand onto the board to form a

valid line, following the color/shape matching rules. Once tiles are placed, the current player scores points for any lines formed and draws new tiles to replenish their hand to six.

In each round, a player may choose not to place any tile on the board and instead exchange some of their tiles for new ones drawn from the bag. Although this move is allowed in natural gameplay, it is not considered in our project. Instead, we assume that every player places at least one tile on the board during each turn.

Scoring

The score of the current player after each move is based on the tiles placed on the board and can increase in some cases.

When a player places tiles, they score for each *newly formed* or *extended* line:

- **line score.** Each line formed or extended is worth a number of points equal to the number of tiles in that line (including newly placed tiles).
- **Qwirkle bonus.** If a line reaches six tiles (the maximum possible of tiles without repetitions), that line is called a *Qwirkle*, which awards a bonus of six additional points.
- **bonus squares.** Placing a tile on a marked bonus square modifies the base scoring for that line — by adding extra points (+1 or +2 points), depending on the square.

If multiple bonuses apply to the same newly placed tile (for example, when creating a *Qwirkle* using bonus squares), their effects are cumulative. For a numbered square (either with 1 or 2), bonus points can be awarded only once, even if the tile placed there belongs simultaneously to two lines and contributes to both.

We consider a line to be composed of at least two tiles (a single tile doesn't form a line).

Scoring example

Figure 5 shows seven moves made by the two players from the initial configuration. We list below the seven moves and offer detailed explanations about computing the corresponding score of each player after each move.

Move 1. Player 1 places three green tiles — a square, a four-point star, and a clover — to build three lines:

- a line of four green tiles, each with a different shape.
- a line of two four-point stars, each in a different color.
- a line of two clovers, each in a different color.

For this move, Player 1 earns 10 points: 4 points for the first line, 2 points each for the second and third lines, and an additional 2 bonus points for placing two tiles on +1 bonus

squares.

Move 2. Player 2 places three red tiles — a four-star, a circle, and a diamond — to form two lines:

- a line of four red tiles, each featuring a unique shape.
- a line of two diamonds in different colors.

This move earns Player 2 a total of 7 points: 4 points for the first line, 2 points for the second line, and an additional bonus point for placing a tile on a +1 bonus square.

Move 3. Player 1 places four orange tiles — a square, a four-point star, a clover, and a circle — to complete two lines:

- a line of five orange tiles, each with a distinct shape.
- a line of two circles in different colors.

This move earns Player 1 a total of 8 points: 5 points for the first line, 2 points for the second line, and 1 bonus point for placing a tile on a +1 bonus square.

Move 4. Player 2 places three orange tiles — an eight-star, a four-star, and a clover — to form two lines:

- a line of six orange tiles, each with a unique shape, thus forming a *Qwirkle*.
- a line of three orange tiles, each with a unique shape.

This move earns Player 2 a total of 17 points: 6 points for the first line plus an additional 6-point bonus for completing a *Qwirkle*, 3 points for the second line, and 2 bonus points for placing a tile on a +2 bonus square.

Move 5. Player 1 places three white tiles — an eight-star, a diamond, and a circle — to form two lines:

- a line of four white tiles, each with a distinct shape.
- a line of two circles in different colors.

This move earns Player 1 a total of 7 points: 4 points for the first line, 2 points for the second line, and 1 bonus point for placing a tile on a +1 bonus square.

Move 6. Player 2 places four circle tiles — a blue, a yellow, a green, and an orange one — to form three lines:

- a line of six circle tiles in different colors, thereby forming a *Qwirkle*.
- a line of two yellow tiles featuring different shapes.

- a line of two green tiles featuring different shapes.

This move earns Player 2 a total of 18 points: 6 points for the first line plus an additional 6-point bonus for completing a *Qwirkle*, 2 points for the second line, 2 points for the third line, and 2 bonus points for placing two tiles on +1 bonus squares.

Move 7. Player 1 places two white tiles — a four-point star and a clover — to extend a line from four to six tiles, thereby completing a *Qwirkle*. For this move, Player 1 earns 12 points in total: 6 points for the extended line and an additional 6 points for achieving a *Qwirkle*.

Data description

The release data directory (available here https://tinyurl.com/CV-2025-Project2) contains four directories: *board+tiles*, *train*, *test* and *evaluation*. All directories contain images taken with Bogdan's phone. The directory *board+tiles* contains several images with the empty board from different viewpoints and also with the board on which we placed all possible tiles. You can use these images to better understand the problem and to extract data for your solution. The directories *train* and *test* have the same structure, although the *test* data will be made available after the deadline. In total there are 105 images and 100 annotation files. The training image i corresponding to game g is denoted by the file 'g_i.jpg', where $g \in \{1, 2, 3, 4, 5\}$ and $i \in \{00, 01, 02, 03, ..., 20\}$. In each game the image 'g_00.jpg' corresponds to the configuration of the board after the initial stage, when Player 1 chooses the arrangement of the four quadrants and places 24 tiles on the black squares. The corresponding annotation file has the extension '.jpg' replaced by '.txt'. Notice that the file 'g_00.txt' doesn't exist.

All games consist of images taken from different viewpoints: (i) regular view - the images are taken with the phone placed parallel above the board, with minor scale changes and rotations; (ii) rotated view - the images are taken with the phone placed parallel above the board, but at some non-negligible rotation with respect to the board; (iii) perspective view - the images are taken with the phone tilted with respect to the board.

The ground-truth annotation files contain, for each move, the following information:

- the position of the tile placed on the board, taken in the order from left to right and from top to down. We specify the position using numbers 1-16 for rows and letters A-P for columns.
- a number from 1-6 for shapes (circle - 1, clover - 2, diamond-3, square-4, four-point star - 5 and eight-point star - 6) and a letter for color (red - R, blue - B, green - G, yellow -Y, orange - O and white - W) on the tile at the corresponding position on the board.
- the score for the move.

Figure 6 exemplifies a ground-truth annotation file for image 1_01.jpg

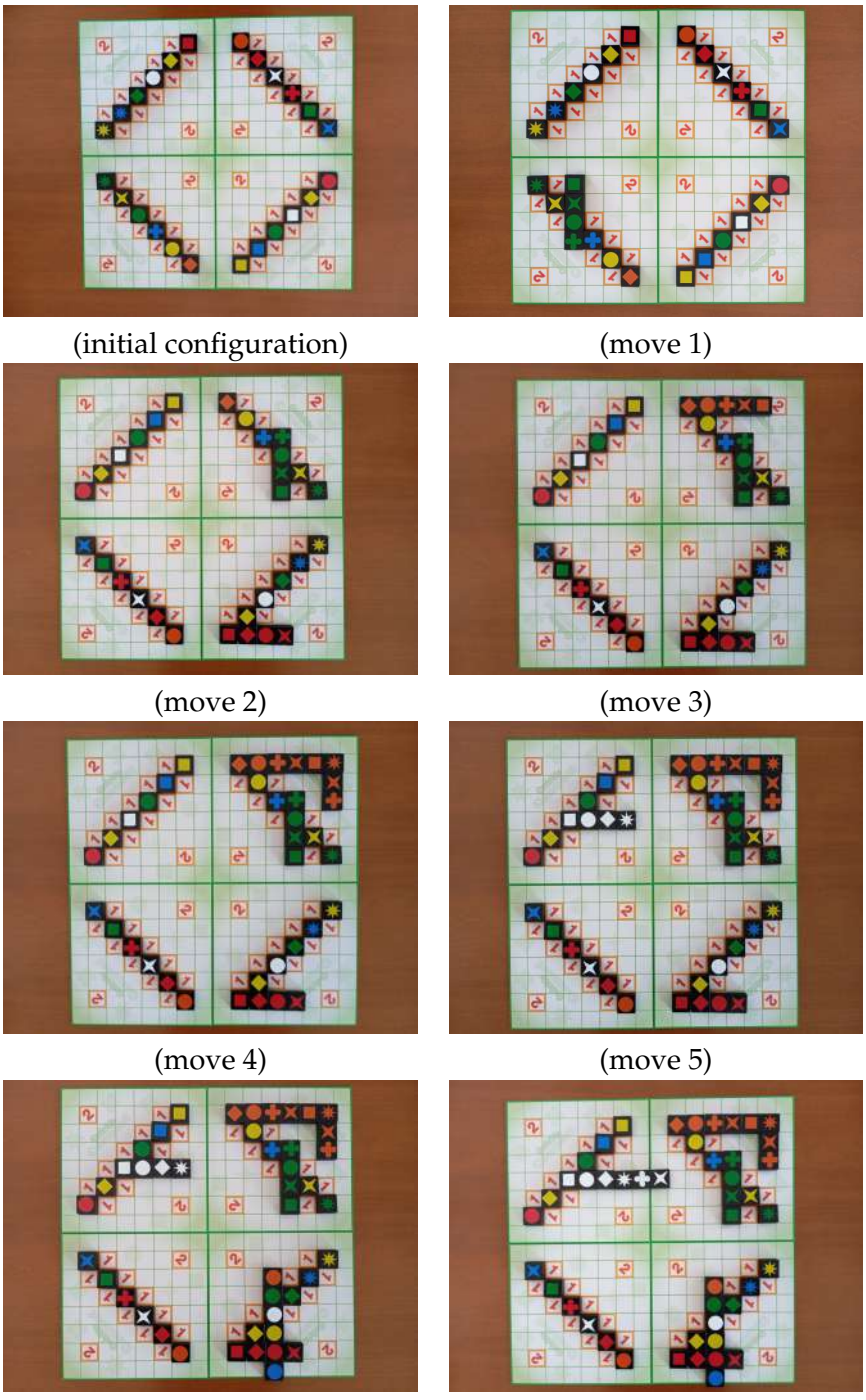


Figure 5: *We illustrate seven moves made by the two players, starting from some initial configuration. We describe in detail in the main text the computation of the corresponding scores to each of the two players after each move.*

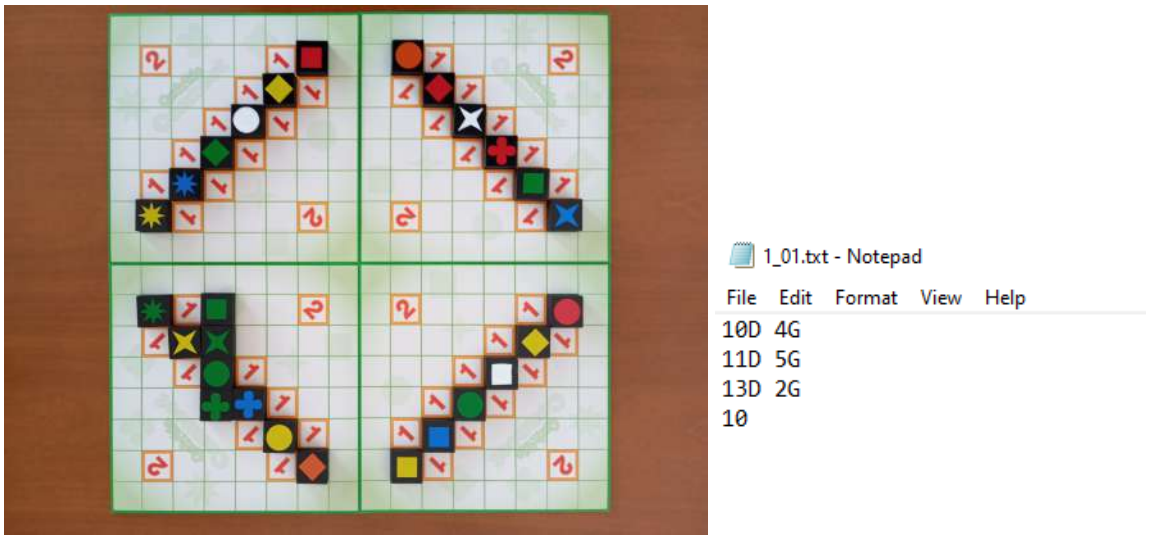


Figure 6: *For the current move, three tiles are placed on the board. The ground-truth annotation file specifies the positions of newly added tiles and the specific color and shape encoded by digits and letters.*

The directory *evaluation* shows how the evaluation will take place on the test data after the deadline. It contains the following subdirectories:

- *fake_test* - this directory exemplifies how the test data will be released, keeping the structure of the previously described *train* directory. Notice that we include here only 1 game = 21 images. The test data will contain 5 games = 105 images.
- *submission_files* - this directory exemplifies the format of the results data that we expect from you to submit in the second stage. You will have to send your results in this format, uploading a zip archive of a folder similar with the one called *Alexe_Bogdan_407*. Notice that the current archive correspond to the data released in the *fake_test* directory. For the test stage your archive should contain around 100 files.
- *code* - this directory contains code that we will use to evaluate your results using the ground-truth data. Make sure that this code will run on your submitted .txt files. The ground-truth data will be released after you send us your results.

Requirements

Your job is to write a program in Python that automatically solves the task of extracting information of the current move depicted in a test image. For each test image (except the first one named 'g_00.jpg'), you have to output the corresponding information similar to the annotation files, thus specifying the position of the newly added tiles to the board, the tiles, and the score.

This project is worth 5.5 points (5 points for regular tasks and 0.5 points for bonus tasks - discussed later). We will grade your project based on the performance achieved by your

algorithms on each of 100 test images (105 total images - 5 initial images in each game).

You will receive a test set containing 105 testing images organized in 5 games of 21 moves. The distribution of images in the test data follows the distribution of train data, meaning that the images were acquired in the same conditions. For each test image you have to output a .txt file containing information similar to the annotation files. Each correctly solved test image is worth 0.04 points. For correctly specifying the position of the added tiles you receive 0.02 points per image (move), for correctly specifying the tiles placed at the corresponding positions on the board you receive 0.015 points per image (move) and for correctly specifying the score of the current player after a move you receive 0.005 points. You receive 0.5 points from *ex officio* conditioned on the fact that you respect the format of the submitted results, such that our evaluation script works smoothly on your provided results.

The oral presentation of this project (face-to-face or online) will be scheduled in the week $6th - 9th$ of May. It will take around 15 minutes in which Alexandra or Bogdan will ask questions regarding implementation. The oral presentation will count for 0.5 points and is mandatory for each student submitting their solution for this project.

Bonus

The bonus scenario addresses the problem of determining the positions of the newly added tiles, recognising the tiles and computing the score at each turn when playing without a board. In this scenario, gameplay occurs directly on the table, without a predefined grid or coordinate system.

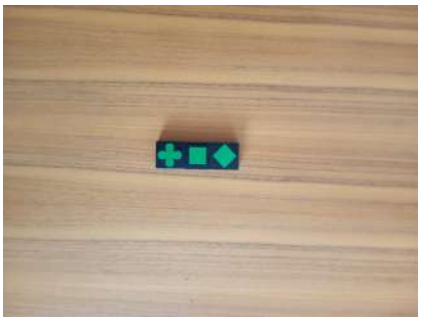
Since there is no board, there is no natural coordinate frame, and specifying tile positions becomes ill-posed. To resolve this, we define the origin $(0, 0)$ as the position of the first tile placed during the initial turn — specifically, the leftmost tile in the top row. All subsequent tile positions are computed relative to this origin. Note that, in this system, negative coordinates may occur.

The method for specifying tiles using digits and letters remains unchanged from the regular Connect scenario. In terms of scoring, the bonus squares that grant +1 or +2 points are removed; however, the 6-point bonus for achieving a *Qwirkle* is still awarded.

Bonus example

Figure 7 shows eight moves made by the two players in the bonus scenario. We list below the eight moves and offer detailed explanations about computing the positions of the tiles placed on the surface and the corresponding score of each player after each move.

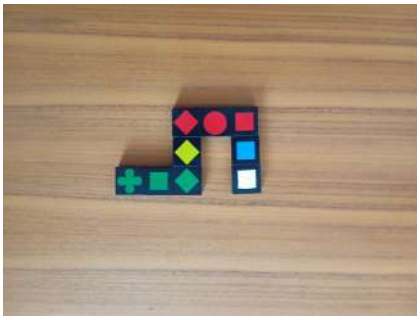
Move 1. Player 1 places three green tiles - a clover, a square, and a diamond - on the empty surface. Their positions are defined as follows:



(move 1)

(move 2)

(move 3)



(move 4)

(move 5)

(move 6)

(move 7)

(move 8)

Figure 7: *We illustrate eight moves made by the two players, starting from some initial configuration. We describe in detail in the main text the computation of the corresponding scores to each of the two players after each move.*

- Column 0, Row 0: green clover;
- Column 1, Row 0: green square;
- Column 2, Row 0: green diamond.

Player 1 earns 3 points for this move.

Move 2. Player 2 places two diamond tiles of different colors - yellow and red - on the surface. Their positions are defined as follows:

- Column 2, Row 2: red diamond;
- Column 2, Row 1: yellow diamond;

Player 2 earns 3 points for this move.

Move 3. Player 1 places two red tiles of different shape - circle and square - on the surface. Their positions are defined as follows:

- Column 3, Row 2: red circle;
- Column 4, Row 2: red square;

Player 1 earns 3 points for this move.

Move 4. Player 2 places two square tiles of different colors - blue and white - on the surface. Their positions are defined as follows:

- Column 4, Row 1: blue square;
- Column 4, Row 0: white square;

Player 2 earns 3 points for this move.

Move 5. Player 1 places two blue tiles of different shape - four-point star and eight-point star - on the surface. Their positions are defined as follows:

- Column 5, Row 1: blue four-point star;
- Column 6, Row 1: blue eight-point star;

Player 1 earns 3 points for this move.

Move 6. Player 2 places two clover tiles of different colors - yellow and orange - on the surface. Their positions are defined as follows:

- Column 0, Row -1: yellow clover;
- Column 0, Row -2: orange colver;

Player 2 earns 3 points for this move.

Move 7. Player 1 places two orange tiles of different shape - eight-point star and circle - on the surface. Their positions are defined as follows:

- Column -2, Row -2: orange eight-point star;
- Column -1, Row -2: orange circle;

Player 1 earns 3 points for this move.

Move 8. Player 2 places two orange tiles of different shape - square and diamond - on the surface. Their positions are defined as follows:

- Column 1, Row -2: orange square;
- Column 2, Row -2: orange diamond;

Player 2 earns 5 points for this move.

Deadlines

Submit a *zip archive* containing your code (Python files or Jupyter notebook files), all auxiliary data that you are using (templates, models, etc.) and a pdf file describing your approach until Monday, 5^{th} of May, 11.59 PM using the following link https://tinyurl.com/CV-2025-PROJECT1-SUBMISSIONS. Please do not include in your zip archive any unuseful data (like training images, we already have them!!!). Notice that this is a hard deadline, no projects will be accepted after the deadline. Your code should include a README file (see the example in the materials for this project) containing the following information: (i) the libraries required to run the project including the full version of each library; (ii) indications of how to run the solution and where to look for the output file. Students who do not describe their approach (using a pdf file) will incur a penalty of 0.5 points.

On Tuesday 6^{th} of May we will make available the test data. You will have to run your solution on the test images provided by us and upload your results in the same day as a zip archive using the following link https://tinyurl.com/CV-2025-PROJECT1-RESULTS.

IMPORTANT NOTE. After the deadline, you are not allowed to make any changes that improve your solution on the test data. This includes modifying code parameters, updating trained models, altering templates, or similar adjustments. Please ensure that you test your solution in a newly created environment following your README instructions to confirm that it runs correctly on our machines. We will execute your code to verify that your reported results match our outputs. You may only change non-critical settings, such as adjusting paths for test data or modifying try/except blocks if your code produces an error when processing a test image.