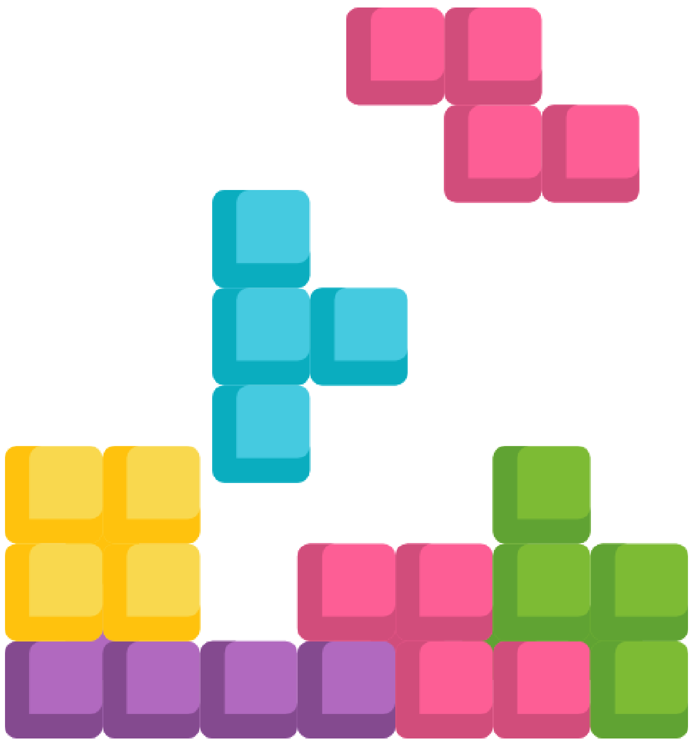
De Saleneuve Adam | S2B’
Lecoeuvre Mathis | S2B’’

Project report

TetroCity

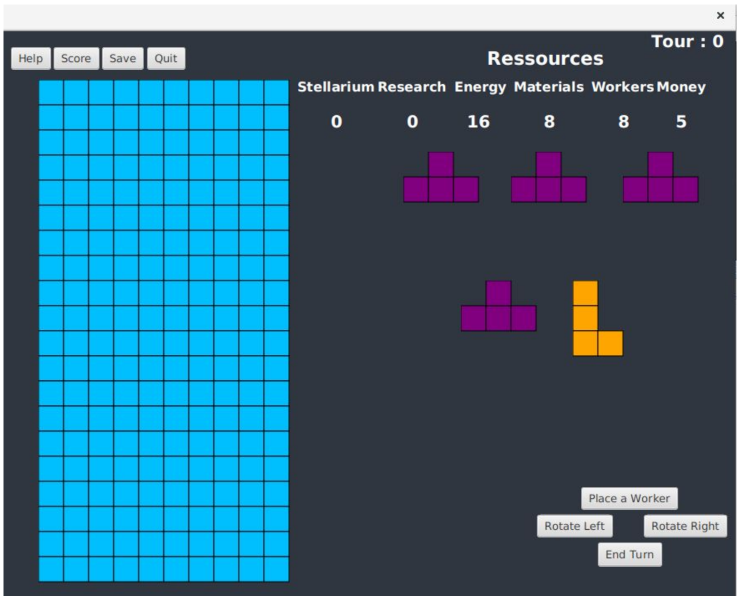


Summary

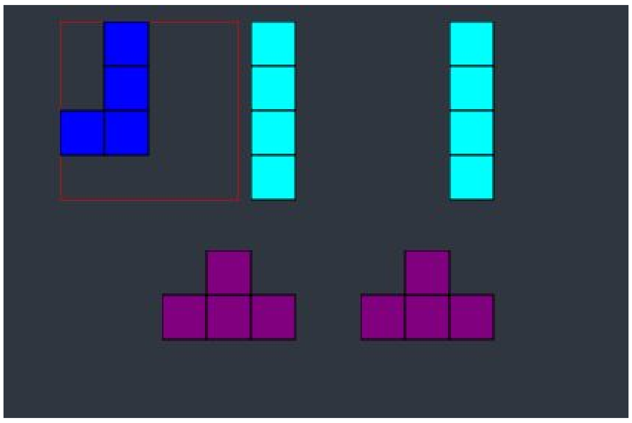
I/Description of final product

Do you like board games? Do you like tetrominos? TetroCity is, for you, the perfect game to have fun with your favourites shapes from Tetris.
TetroCity is a board game which the player will be able to place Tetrominos and workers.

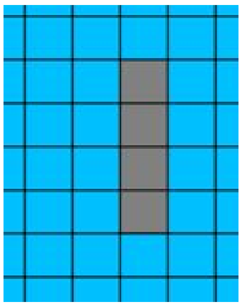
When you start a new game, it will look like that:



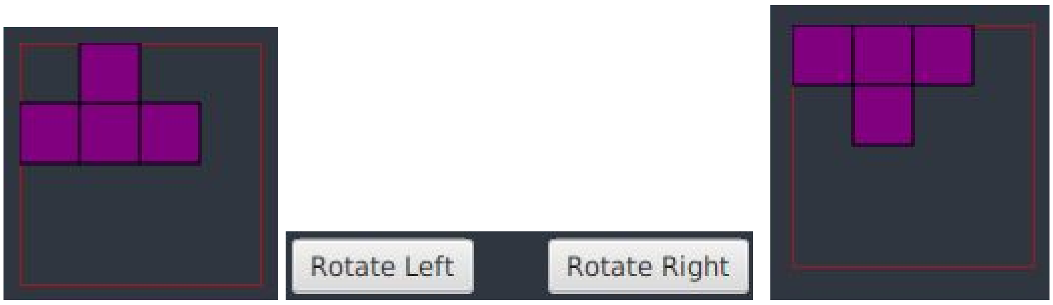
In our game, the player will be able to click on his buildings to select one of them .After clicking, the building will look like that :



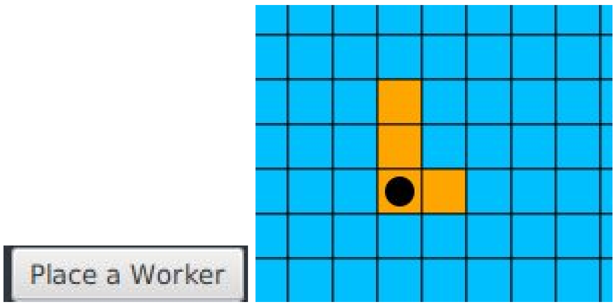
After the player selected a building, he will be able to place it on the board if he has enough resources to build it, the building will also have a preview to see where it will be placed on the board :



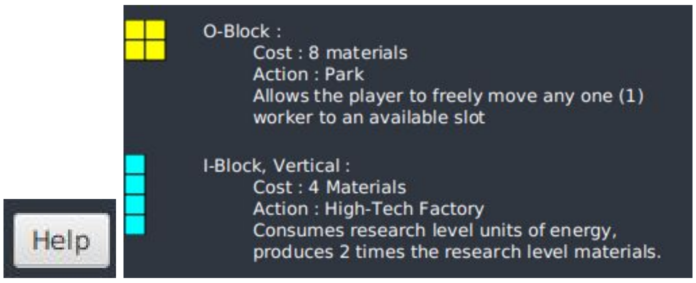
The player can rotate the selected building by clicking on the buttons “rotate left”, “rotate right” :



After that, he will be able to place a worker in a building, by clicking on the button “place worker” then by clicking on a building which is on board to permit him to trigger the associated action at the end of turn :



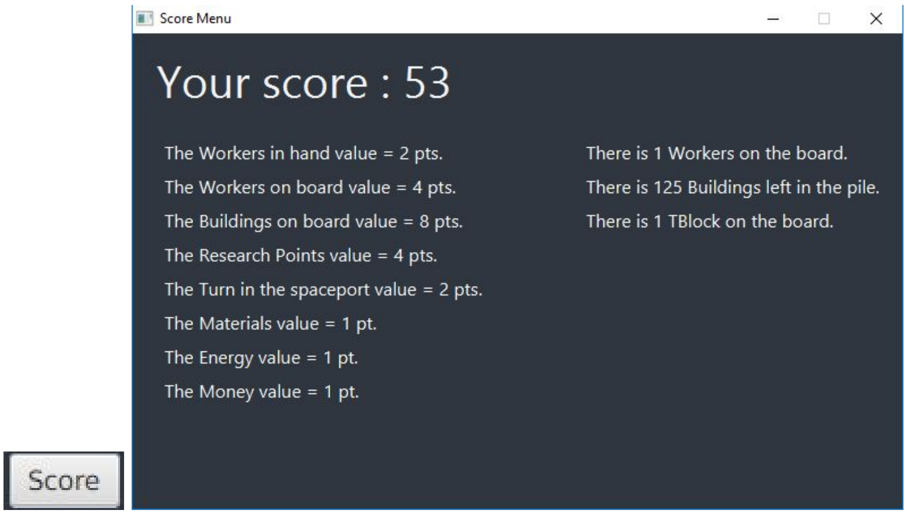
The player can go to the help menu with the help button anytime to see the actions of every buildings in the game and their cost :



O-Block :
Cost : 8 materials
Action : Park
Allows the player to freely move any one (1) worker to an available slot

I-Block, Vertical :
Cost : 4 Materials
Action : High-Tech Factory
Consumes research level units of energy, produces 2 times the research level materials.

The player can go to the score menu anytime to see his score and the value of each resources in the game :


Score Menu

Your score : 53

The Workers in hand value = 2 pts.
The Workers on board value = 4 pts.
The Buildings on board value = 8 pts.
The Research Points value = 4 pts.
The Turn in the spaceport value = 2 pts.
The Materials value = 1 pt.
The Energy value = 1 pt.
The Money value = 1 pt.

There is 1 Workers on the board.
There is 125 Buildings left in the pile.
There is 1 TBlock on the board.

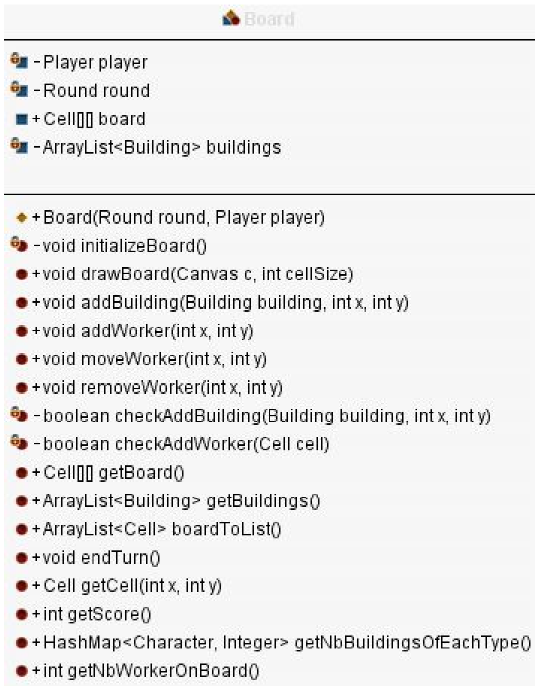
The player can also save his current progression, but our game can only have one saved file at the same time. To continue his game later, he can use the "save" button.

The player can leave the game with the "quit" button :

II/Design and specifications

We have already brought several modifications to our conception since our last report. We were clearly missing a JavaClass that were allowing us to save the informations which were contained in a cell of the board. Our board is now a two entry array. The pile is not instantiated anymore by the round class but by Player as an attribute. This last improvement is allowing us to have the same pile for every players. Therefore the RoundClass is not linked to the others. It is used has a turn counter and as a main function to start JavaFx. The whole part with the buildings hasn't changed. We also added few methods that we will talk about later. We also added an enumeration to give buildings orientations.We had created a package with 2 classes of exceptions to lift exceptions in specials cases. Finally we added multiples classes relative to JavaFx for graphic display including one class for each screen (Menu, Game, Score, Help) .

In the class Board, we made a private method which initialize the cells of the board into Cell. Much more methods were being added to facilitate the graphic display.The methods used to add a building or a worker were moved into this class. We also added an method name endTurn which will trigger the actions of the buildings on the board if they have workers on themselves and which will verify if the player has buildings in his hand to make him draw one or not. We also added some getters to get the access of informations in other classes.

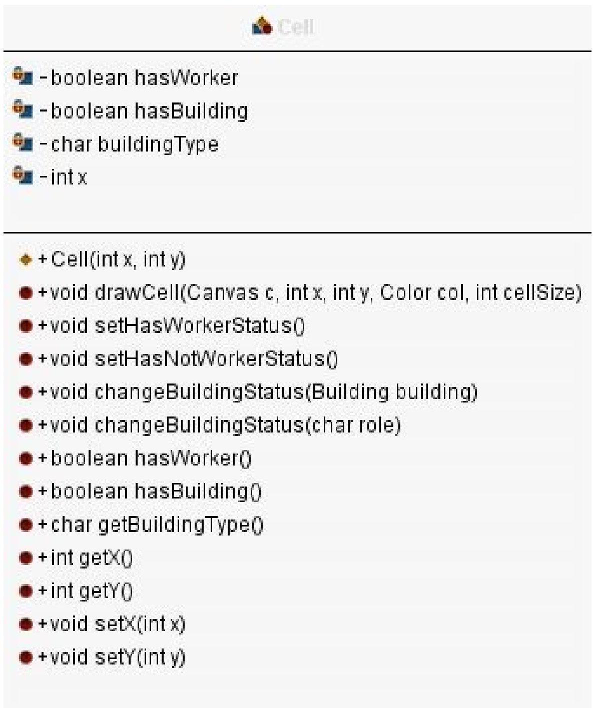


```
Board

-Player player
-Round round
+Cell[][] board
-ArrayList<Building> buildings

+Board(Round round, Player player)
-void initializeBoard()
+void drawBoard(Canvas c, int cellSize)
+void addBuilding(Building building, int x, int y)
+void addWorker(int x, int y)
+void moveWorker(int x, int y)
+void removeWorker(int x, int y)
-boolean checkAddBuilding(Building building, int x, int y)
-boolean checkAddWorker(Cell cell)
+Cell[][] getBoard()
+ArrayList<Building> getBuildings()
+ArrayList<Cell> boardToList()
+void endTurn()
+Cell getCell(int x, int y)
+int getScore()
+HashMap<Character, Integer> getNbBuildingsOfEachType()
+int getNbWorkerOnBoard()
```

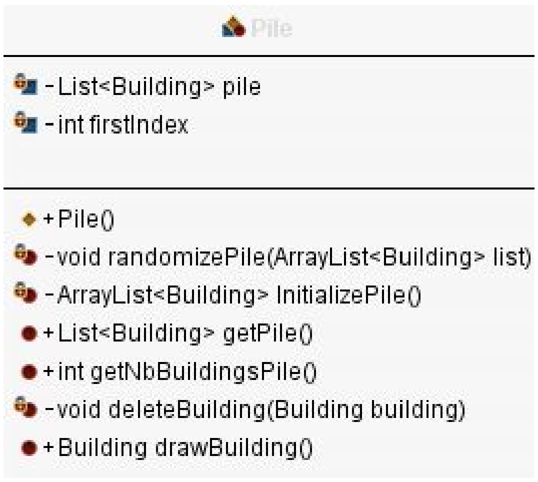
The Cell class has been created and contains a position, a building type and a boolean that defines if he has a worker or not. This class has mainly getters and setters.



```
Cell
-boolean hasWorker
-boolean hasBuilding
-char buildingType
-int x

+Cell(int x, int y)
+void drawCell(Canvas c, int x, int y, Color col, int cellSize)
+void setHasWorkerStatus()
+void setHasNotWorkerStatus()
+void changeBuildingStatus(Building building)
+void changeBuildingStatus(char role)
+boolean hasWorker()
+boolean hasBuilding()
+char getBuildingType()
+int getX()
+int getY()
+void setX(int x)
+void setY(int y)
```

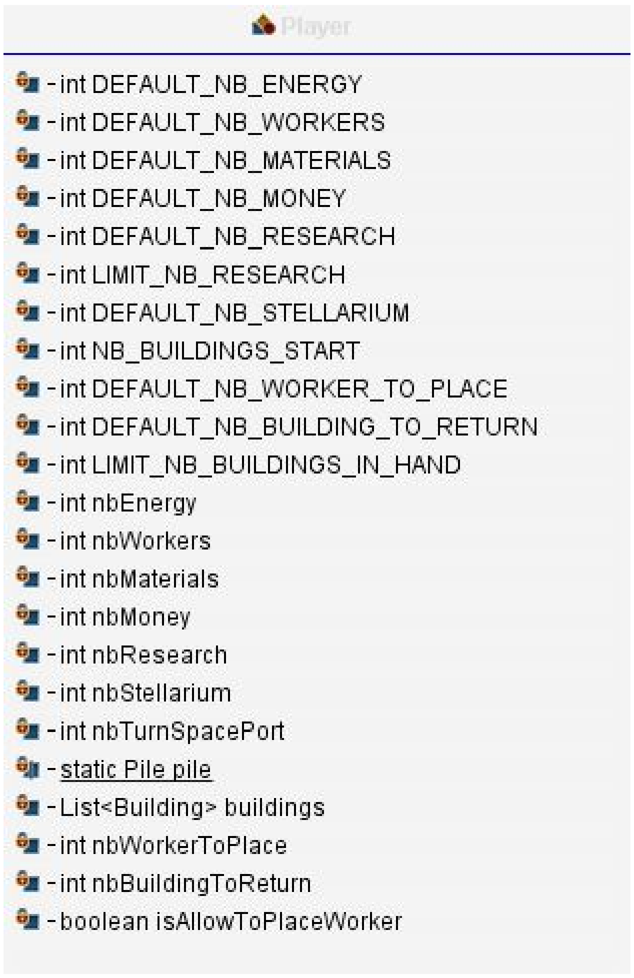
The Pile class stays faithful to our basic design. We have just added methods to remove a building from the pile and draw a building on the top of the pile.



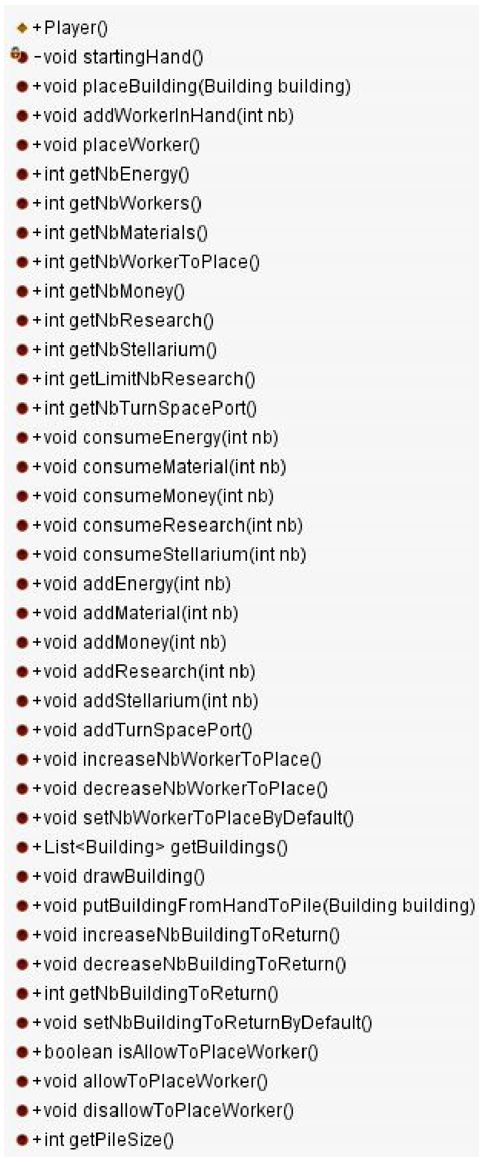
```
Pile
-List<Building> pile
-int firstIndex

+Pile()
-void randomizePile(ArrayList<Building> list)
-ArrayList<Building> InitializePile()
+List<Building> getPile()
+int getNbBuildingsPile()
-void deleteBuilding(Building building)
+Building drawBuilding()
```

For the player class, we have added a first private method that is called in the constructor to draw 5 buildings from the pile from its creation. We have, as well, a method that will deduct the building's cost to the number of materials when he is placed on the board. We have added some methods that add a number to an other number of resources and that remove and some getters for that. There are, as well, a new method that check if the player has less than 5 buildings before draw a new one, else he don't draw. We fixed it for display purposes. We have added a method to add building from the player's hand to the end of the pile to exchange by the first building on the pile.
Finally, we have added a method to permit or not the player to place a worker on the board.

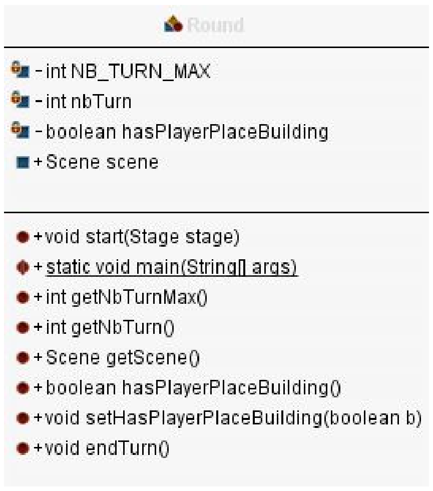


```
Player
- int DEFAULT_NB_ENERGY
- int DEFAULT_NB_WORKERS
- int DEFAULT_NB_MATERIALS
- int DEFAULT_NB_MONEY
- int DEFAULT_NB_RESEARCH
- int LIMIT_NB_RESEARCH
- int DEFAULT_NB_STELLARIUM
- int NB_BUILDINGS_START
- int DEFAULT_NB_WORKER_TO_PLACE
- int DEFAULT_NB_BUILDING_TO_RETURN
- int LIMIT_NB_BUILDINGS_IN_HAND
- int nbEnergy
- int nbWorkers
- int nbMaterials
- int nbMoney
- int nbResearch
- int nbStellarium
- int nbTurnSpacePort
- static Pile pile
- List<Building> buildings
- int nbWorkerToPlace
- int nbBuildingToReturn
- boolean isAllowToPlaceWorker
```



```
+Player()
-void startingHand()
+void placeBuilding(Building building)
+void addWorkerInHand(int nb)
+void placeWorker()
+int getNbEnergy()
+int getNbWorkers()
+int getNbMaterials()
+int getNbWorkerToPlace()
+int getNbMoney()
+int getNbResearch()
+int getNbStellarium()
+int getLimitNbResearch()
+int getNbTurnSpacePort()
+void consumeEnergy(int nb)
+void consumeMaterial(int nb)
+void consumeMoney(int nb)
+void consumeResearch(int nb)
+void consumeStellarium(int nb)
+void addEnergy(int nb)
+void addMaterial(int nb)
+void addMoney(int nb)
+void addResearch(int nb)
+void addStellarium(int nb)
+void addTurnSpacePort()
+void increaseNbWorkerToPlace()
+void decreaseNbWorkerToPlace()
+void setNbWorkerToPlaceByDefault()
+List<Building> getBuildings()
+void drawBuilding()
+void putBuildingFromHandToPile(Building building)
+void increaseNbBuildingToReturn()
+void decreaseNbBuildingToReturn()
+int getNbBuildingToReturn()
+void setNbBuildingToReturnByDefault()
+boolean isAllowToPlaceWorker()
+void allowToPlaceWorker()
+void disallowToPlaceWorker()
+int getPileSize()
```

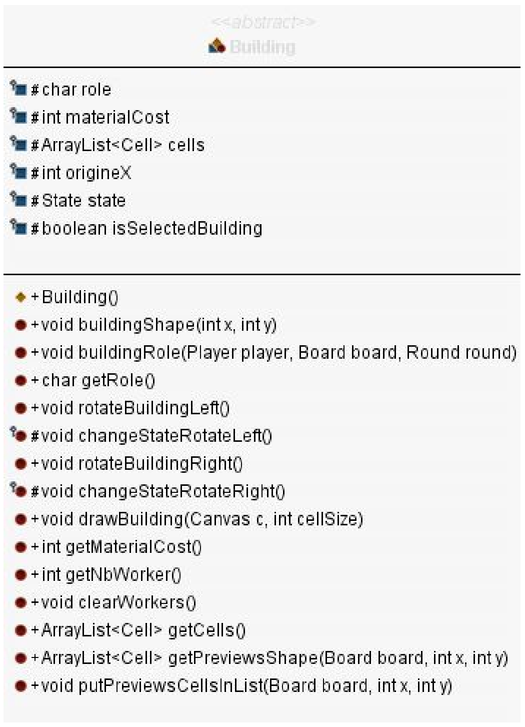
The round class stayed is still the same from our first design report except for an attribute which verify if a player as already placed a building during his turn. We also added a method endTurn which goes up every turn and represent the current turn.


Round

- int NB_TURN_MAX
- int nbTurn
- boolean hasPlayerPlaceBuilding
+Scene scene

+void start(Stage stage)
+static void main(String[] args)
+int getNbTurnMax()
+int getNbTurn()
+Scene getScene()
+boolean hasPlayerPlaceBuilding()
+void setHasPlayerPlaceBuilding(boolean b)
+void endTurn()

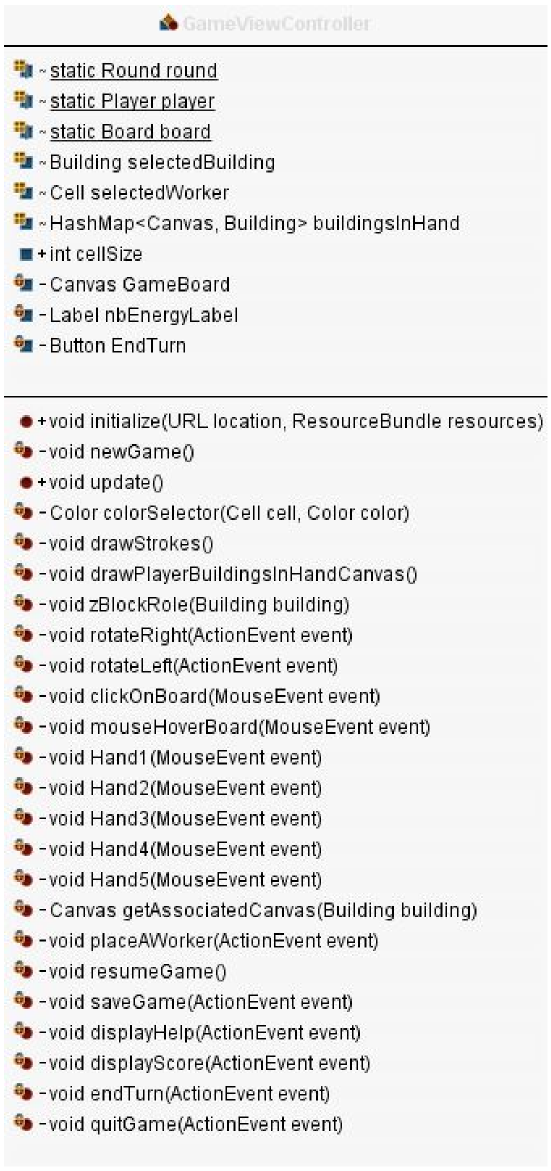
In the abstract class building, we also added an method which will change the orientation of a building during a rotation. We also added getter but also a method which allow the game to put back in hand all the workers of a building if a player doesn't have enough resources to trigger the action. We added a function which return the cells used if a player would place his building in this place, and also another one which puts the cells in a list of cells composing the building.


<<abstract>>
Building

#char role
#int materialCost
#ArrayList<Cell> cells
#int origineX
#State state
#boolean isSelectedBuilding

+Building()
+void buildingShape(int x, int y)
+void buildingRole(Player player, Board board, Round round)
+char getRole()
+void rotateBuildingLeft()
#void changeStateRotateLeft()
+void rotateBuildingRight()
#void changeStateRotateRight()
+void drawBuilding(Canvas c, int cellSize)
+int getMaterialCost()
+int getNbWorker()
+void clearWorkers()
+ArrayList<Cell> getCells()
+ArrayList<Cell> getPreviewsShape(Board board, int x, int y)
+void putPreviewsCellsInList(Board board, int x, int y)

In the GameViewController Class we have all the method which interact in a graphic context. An function which will create a new game or resume a saved one depending on the user's choices. An method named update() is used to update all the graphic components. There is also an method allowing us to define the color of an cell depending on his building type. We also have multiples methods to place and display the graphic components. We also included the method which ZBlock's role in the class. This choice was necessary to have an better clarity of code. We added there 2 methods for the save and reading part. We decided to use the Serialization which a Javamethod which allow us to save objects in a file.



```
GameViewController

~static Round round
~static Player player
~static Board board
~Building selectedBuilding
~Cell selectedWorker
~HashMap<Canvas, Building> buildingsInHand
+int cellSize
-Canvas GameBoard
-Label nbEnergyLabel
-Button EndTurn

+void initialize(URL location, ResourceBundle resources)
-void newGame()
+void update()
-Color colorSelector(Cell cell, Color color)
-void drawStrokes()
-void drawPlayerBuildingsInHandCanvas()
-void zBlockRole(Building building)
-void rotateRight(ActionEvent event)
-void rotateLeft(ActionEvent event)
-void clickOnBoard(MouseEvent event)
-void mouseHoverBoard(MouseEvent event)
-void Hand1(MouseEvent event)
-void Hand2(MouseEvent event)
-void Hand3(MouseEvent event)
-void Hand4(MouseEvent event)
-void Hand5(MouseEvent event)
-Canvas getAssociatedCanvas(Building building)
-void placeAWorker(ActionEvent event)
-void resumeGame()
-void saveGame(ActionEvent event)
-void displayHelp(ActionEvent event)
-void displayScore(ActionEvent event)
-void endTurn(ActionEvent event)
-void quitGame(ActionEvent event)
```

The other graphic classes just serve to display the differents screens.

Here the classes roles:

- Board : Store the information about the board and allow to edit their.
- Cell : Store the information about a cell of the board.
- Pile : Store buildings in the pile and allow to draw one.
- Player : Store information about player’s resources and allow to edit their
- Round : Serves as a turn counter and application launcher.
- Building : Store information and properties about a building and allow to return its.
- State (enumeration) : Serves to specify building’s orientation.
- InsufficientRessourcesException : Allow to launch an exception when the player has not enough resources to trigger action.
- InvalidLocationException : Allow to launch an exception when the player want to place an entity on an invalid location.
- GameViewController : Display and Update the graphic context during the game.
- MenuViewController : Manage the display of the Menu.
- HelpViewController : Manage the display of the HelpMenu.
- ScoreViewController : Manage the display of the Score.

III/Realisation evaluation

There are some elements that we regret, we tried to do our best to optimize our code. Despite this, we could have rectified some points, especially ZBlock’s role which is stored in GameViewController class. Our code is probably missing some refactoring and has redundant code. In the Building’s daughter classes, we have puting hard values to the graphic organisation of cells, in rotation methods in particular. Our choice is explained by the fact these values will make the code less readable if they were stored in attributes.

We have tried to realize the first possible feature but unfortunately, we doesn't have to lead it. Indeed, have implemented all the new functionality skips that the player can use stellarium points. In addition to what is expected, we have implemented a help screen to permit the player to find his way between buildings’ differents roles and the cost of construction and production.

If we had to start from scratch again, we’d have to think how to organize ourselves in vertical and horizontal axes and set postulates. We had a lot of problems because of this because from a graphic point of view, x is horizontal axe and y is vertical axe. On the other hand, from a logic point of view (with tables), it’s the opposite, x axe is a row and y axe is a column. We would also have used an separate class to the “main” method instead of grafting it into Round class.

IV/Work organisation

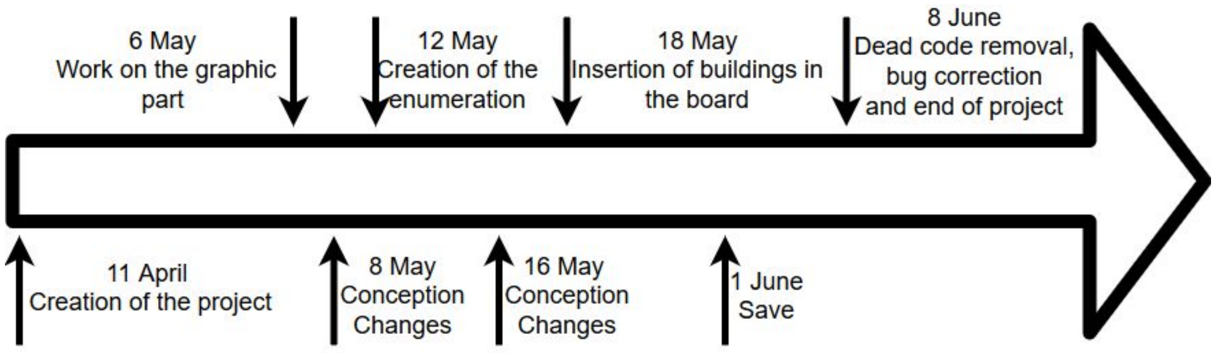
Adam :

Graphic part
Creation of the screens for the display
Managed the graphic entity’s movements on the screens
Managed the errors
Contributed to the logic part

Mathis:

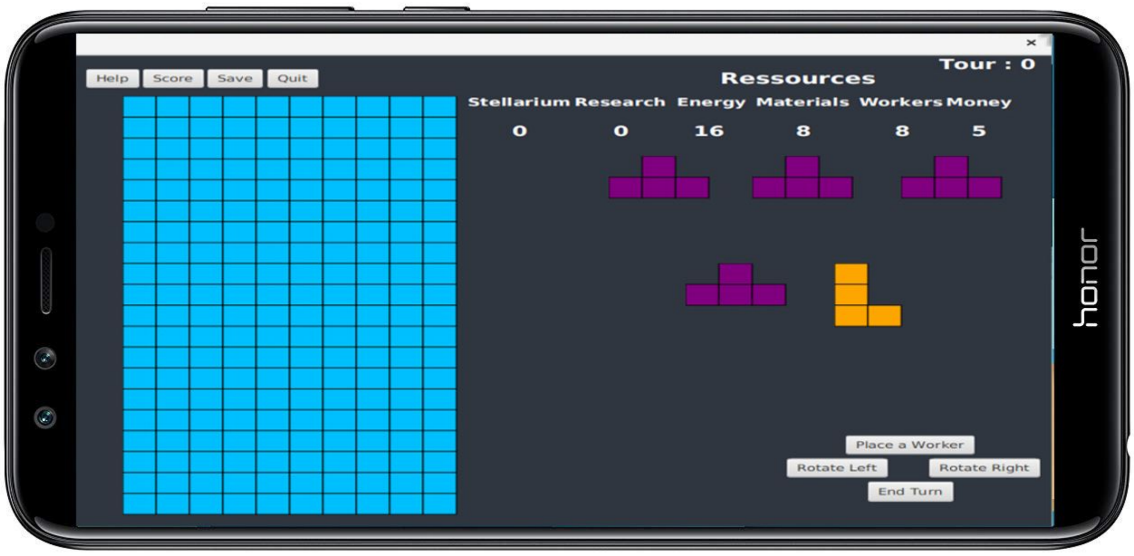
Logic part
Creation of the logic classes
Linked the classes between them
Established the game logic
Contributed to the graphic part

11 April: Creation of the project, packages, classes
6 May: Work on the graphic part
8 May: Conception changes (Pile, Board et Player seront géré par Round)
12 May: Creation of the enumeration for the block’s orientation
16 May: Conception changes (Pile est géré par Player)
18 May: Insertion of buildings in the board
1 June: Save
8 June: Dead code removal, bug correction



V/Perspectives

If we had a budget, time and human resources to finalize this project we would have finished the game correctly, for example avoiding certains bugs and implementing correctly the stellarium. We would also have tried to do like 3 or 4 more features like adding more players or adding balance changes to make the game more playable in most situation like to avoid starting with an hand of 5 buildings which each cost 8 materials and doesn't produces any. We would also tried to export it on mobile to make it playable everywhere, it would have looked like this :



We would do better graphics, a carousel to have more than 5 buildings in hand, optimisations and some refactoring.