Logo

Description automatically generated

OBJECT – ORIENTED PROGRAMMING PROJECT

TETRIS GAME

le thi huynh giao ititiu19012

LA TRI NGUYEN ITITIU19036

pham ha minh thy ititiu19056

NGO BINH PHUONG NGHI ITITIU10163

INTERNATIONAL UNIVERSITY

VIETNAM NATIONAL UNIVERSITY

Table of Contents

**introduction 2**

**PROPERTIES OF TETRIS 2**

1. **GOAL 2**
2. **RULE 3**

**THEORY 3**

1. **CLASS 3**
2. **OBJECT 3**

Object Declaration

1. **OTHER FUNCTIONS 4**
2. Create Game Looper
3. Load Image
4. Load Sound
5. Draw Image
6. Set Font
7. Set Color
8. Key Pressed
9. Mouse Pressed
10. Mouse Released

**UML DIAGRAM** **7**

**METHODOLOGY 8**

**MAIN CLASS – FUNCTION – EXPLAIN 10**

1. **SHAPE CLASS 10**
2. Collision Checking Algorithm
3. Line Checking Algorithm
4. Rotate Algotithm
5. **BOARD CLASS 13**

Shape Moving Algorithm

**RESULT – LIMIT – CONCLUSION 19**

1. **RESULT**
2. **LIMIT**
3. **CONCLUSION**

# 

# **INTRODUCTION**

These days, with the rapid development of Software Technology industry, the high level in programming skill becomes more necessary. Therefore, classic Procedural – Oriented Programming method cannot be applied for all requirements. This problem leads to the invention of a new method following the ***Alan – Kay*** principle named **“Object – Oriented Programming”**.

This project was designed based on Object – Oriented Programming method by Java language. This method has solved some problems occurred in the building process using Procedural – Oriented Method:

* The code is clear, concise and easy to understand.
* The project is a unified logic system of related classes.
* Each class has methods which take on different behaviors.
* The ability of reusing the resources.

The purpose of this project is to design a basic game that was built on the foundation of object – oriented. Our group decided to modify a game named “Tetris”, following the principle of this method. Tetris is a game which requires the user to arrange different shapes of cubes in order that there will be no spaces in a line. This game will clearly show the object – oriented properties, the combinations of classes and the relationships between classes and objects.

Being inspired by this famous game, our group choose to create a special version of Tetris by applying the knowledge that we have learnt from the court Object-Oriented Programing. Furthermore, we want to challenge and improve ourselves on building a software that runs smoothly. In the process of creating, we have met many difficulties that force us to practice our soft skills, such as teamwork, problem solving, information finding and selecting. Through this project, we have found ourselves getting better and having a deeper understanding of the object-oriented programing’s theory.

1. **PROPERTIES OF TETRIS**
2. **Goal**

Blocks fall down from the top of the screen. They can be rotated, moved to the left or to the right. Blocks fall at the normal rate, but player can make them fall faster if the position is assured. The objective is to get the blocks to fill all empty spaces of one or more lines; whenever this happens, blocks will vanish, and points will be earned.

1. **Rule**

Tetris has very simple rules. Pieces can only be rotated and moved in some specific ways: left, right, down. Game is over whenever your pieces reach the top of the screen. Pieces can only be removed from the screen by filling all the blank spaces in a line. 10 points will be earned whenever a line is removed. Our game has 3 levels. When player reach 30 points, they will move to the next level.

1. **THEORY**
2. **Class**

In Object – Oriented Programming, a **class** is a blueprint for creating **objects** (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods).

The user – defined objects are created using ***class*** keyword. The class is a blueprint that defines a nature of a future object. An **instance** is a specific object created from a particular class. Classes are used to create and manage new objects and support **inheritance** – a key ingredient in object – oriented programming and a mechanism of reusing code.

*Class Declaration*

**public class** < class name > {

<list of instance variables>

<list of methods>

}

1. **Object**

An Object is nothing but a self – contained component which consists of methods and properties to make a particular type of data useful. Object determines the behavior of the class. When you send a message to an object, you are asking the object to invoke or excute one of its method.

From a programming point of view, an object can be a data structure, a variable or a function. It has a memory location allocated. The object is designed as class hierarchies.

### *Object Declaration*

ClassName ReferenceVariable = **new** ClassName ();

Diagram, timeline

Description automatically generated

1. **Other functions**
2. **Create game looper**

VariableName = **new** Timer (delay, new ActionListener() {

@Override

Public void actionPerformed(ActionEvent e) {

<list of functions>

}

});

1. **Load Image**

public class **ImageLoader** {

public static BufferedImage loadImage (String path) {  **try** {

return ImageIO.read(**ImageLoader**.class.getResource(path));

} **catch** (IOException e) {

e.printStackTrace();

System.exit(1);

}

return null; }}

1. **Load Sound**

public class **ImageLoader** {

public static Clip LoadSound (String path) {

**try** {

Clip clip = AudioSystem.getClip(); clip.open(AudioSystem.getAudioInputStream(ImageLoader.class.getResource(direction)));

return clip;

} **catch** (IOException e) {

e.printStackTrace();

}

return null; }}

1. **Draw Image**

boolean **Graphics**.drawImage(Image img, int x, int y, ImageObserver observer);

1. **Set Font**

boolean **Graphics**.setFont ( new Font (<name>, Font.<Style>, <size>));

1. **Set Color**

boolean **Graphics**.setColor(Color.<color>);

1. **Key Pressed**

public class <NameClass> extends JPanel implements KeyListener {

@Override

public void keyPressed (KeyEvent e) {

if (e.getKeyCode () ==KeyEvent.VK\_LEFT) <functions>;

if (e.getKeyCode () ==KeyEvent.VK\_RIGHT) <functions>;

if (e.getKeyCode () ==KeyEvent.VK\_DOWN) <functions>;

if (e.getKeyCode () ==KeyEvent.VK\_UP) <functions>;

}}

1. **Mouse Pressed**

public class <NameClass> extends JPanel implements MouseListener {

@Override

public void mousePressed (MouseEvent e) {

if (e.getButton () == MouseEvent.BUTTON1) leftClick = true;

}}

1. **Mouse Released**

public class <NameClass> extends JPanel implements MouseListener {

@Override

public void mouseReleased (MouseEvent e) {

if (e.getButton () == MouseEvent.BUTTON1)

leftClick = false;

}}

1. **Mouse Dragged**

public class <NameClass> extends JPanel implements MouseListener {

@Override

public void mouseDraggered (MouseEvent e) {

data members;

}}

1. **Mouse Move**

public class <NameClass> extends JPanel implements MouseListener {

@Override

public void mouseMoved (MouseEvent e) {

data members;

}}

1. **Mouse Entered**

public class <NameClass> extends JPanel implements MouseListener {

@Override

public void mouseEntered (MouseEvent e) {

}}

1. **Mouse Excited**

public class <NameClass> extends JPanel implements MouseListener {

@Override

public void mouseDraggered (MouseEvent e) {

}}

1. **UML DIAGRAM**

Diagram

Description automatically generated

1. **METHODOLOGY**

This Object – Oriented Programming Project, specifically, Tetris Game, is made up of 5 classes named Window, Board, Shape, ImageLoader, and title. The game is built based on our knowledge of object oriented method, therefore, we pay full attention to Shape and Board classes - 2 vital classes of our project.

To begin with, Shape class was created with its properties and behaviors. Each object inherited from Shape can reuse these behaviors. In particular, Shape class can rotate, render, update, etc.

Next, we built class named Board that inherit parent class, Shape, and some Interface classes to form the Tetris Game, like controlling blocks, adding background and increasing score. It performs many important functions:

* Store the current state of background board, start - stop buttons, random the next blocks’ shapes.
* Raise player’s score if the block fall successfully.
* Support for the common operation: add pieces to the board, let pieces gradually fall downward, detect various conditions about the board.

Thirdly, we created the title class to attach the player to the game. Title class implements some Interface classes, for example MouseListener and MouseMotionListener, to process mouse events, like press, click, enter, and exit.

Finally, we link all the classes to a logically system and put it into Window class which contains main function to run the game project.

Diagram

Description automatically generated

**Algorithm of Tetris Game**

1. **MAIN CLASS – FUNCTIONS – EXPLAINATION**
2. **Shape Class**

A picture containing timeline

Description automatically generated

Shape Class is created to design and control shapes easily. It is inherited properties and functions from the general shape. We use matrix type with data type int to design the blocks and put them into xy-plane with x, y coordinates.

Every object in the class can be controlled by all functions. These functions are update, render, check line, rotate, transposed, reverse, etc.…

1. update Function

In the function, the algorithm checks the collision of the shape between two shapes or shape with edges of the board, set the next shape, and check if the line is full.

In the check collision method, we check if the pieces are prevented from moving by other pieces or the edges of the board.

If the value position of the current piece has the same value position with any previous piece or two moving pieces have the same value position of any block, or the pieces reach the limit of the board, a piece cannot move anyway and stop at that coordinate. Then the collision returns false.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. rotateShape Function

The algorithm provides a way for clients to access various piece rotations. The client can ask a pointer piece to the "next rotation" which means yields a pointer piece by the next rotation piece. For example, with J – shape {{1, 1, 1}, {0, 0, 1}}, when rotating it, this shape become L-shape {{1, 1, 1}, {1, 0, 0}}.

In the rotate method, the main algorithm transposes the matrix of the piece. It means we rotate 90 degrees of this piece that makes the matrix change follow the transposeMatrix.

Ảnh có chứa văn bản

Mô tả được tạo tự động

To complete the rotate method, we combinate transposeMatrix method with reverseRows together.

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. speedDown Fuction

Ảnh có chứa văn bản

Mô tả được tạo tự động

The function makes the speed of the game faster after each level. At level 0, the speed is set as normal. We use the if algorithm to set the delay which controls the speed of a specific level.

1. **Board Class**

Diagram, schematic, timeline

Description automatically generated

The board includes all basic elements of a Tetris board. The most obvious feature of it is the "grid" – a 2-D array with Booleans data type that stores fully spots. The lower-left corner spot's position is (0,0) with X increasing to the right and Y increasing upwards. Filled spots are represented by a true value in the grid. The shapes are placed on the board in the gird. In the board class, it contains button to control the game, the sound, the score, start game, stop game, specially, most of behaviors of the shapes are showed up on the board.

* 1. Create shapes

Graphical user interface, application

Description automatically generated

A block is performed by the coordinates of 6 squared blocks. Then we color which block has a value equals to 1 to create a complete shape. We note that each piece has its own coordinates. These shapes are:

* I - Shape is {1,1,1,1}
* T - Shape is {{1,1,1}, {0,1,0}}
* L - Shape is {{1,1,1}, {1,0,0}}
* J - Shape is {{1,1,1}, {0,0,1}}
* S - Shape is {{0,1,1}, {1,1,0}}
* Z - Shape is {{1,1,0}, {0,1,1}}
* O - Shape is {{1,1}, {1,1}}

1. update Function

Ảnh có chứa văn bản

Mô tả được tạo tự động

At any time push the button refresh, the game will restart, the level returns 0.

Ảnh có chứa văn bản

Mô tả được tạo tự động

When the player pushes the button pause, the game will stop temporarily. Then, if the player pushes that button again, the game will continue.

1. paintComponent Function

Ảnh có chứa văn bản

Mô tả được tạo tự động

If the level is less than 3 and the score is more than 20, the level will increase by 1.

Ảnh có chứa văn bản

Mô tả được tạo tự động

The code records the score of the player, then save the highest score. When the game ends, the algorithm will let the player see the high score of the game.

1. checkLine Function

Ảnh có chứa văn bản

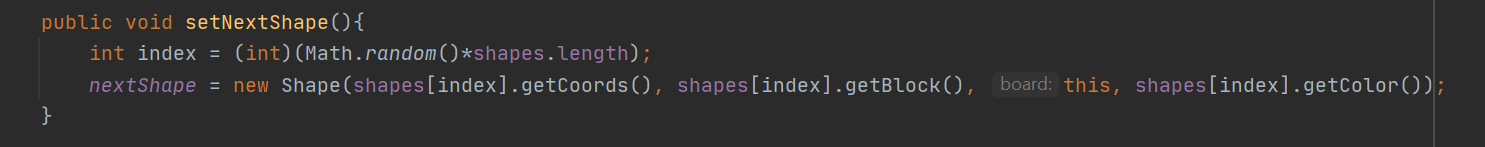
Mô tả được tạo tự động

The check line method is to remove the shape when full of lines. We use the nested if to check each line of the board. If the line is full, remove this line and add more score for the player.

The algorithm in this function is to create a temp line with a coordinate [size][j]. This temp line copies the value of the line that is checked. It is done by the line board.getBoard()[size][j] = board.getBoard()[i][j]. And if each block has a value different from zero, the count increases. When the count equal to 10, the score is added by the code board.addScore(). Then, if the count is not equal to the width of the board, that means the line is not full, then nothing happens, the temp line is the same as the original. On the other hand, if the count is equal to the width, the code “height--” is not working. Therefore, the temp line is not changed, and it copies all values of the next line of the board. It means the full line is alternated by the next line so the algorithm will remove the line and move all lines above fall down one line.

1. setNextShape Function

In this function, the system creates the next shape randomly from default blocks by setNextShape function - (the code setNextShape() will create the next shape randomly from default blocks - I (vertical), J, L, O (Square), S, T, Z) which are created in the class board game.). Then, put that shape in the board game.



1. setCurrentShape

Ảnh có chứa văn bản

Mô tả được tạo tự động

The code makes the game shows the next current shape. When the shape fits in the place the player wants to, at the right side, there will be the next shape for the player.

1. SaveTopScore Function

Ảnh có chứa văn bản

Mô tả được tạo tự động

The function uses the if algorithm to read the score the sort to recognize the highest score of the player. Then the code saves the highest score to output to the screen in the paintComponent function

1. startGame, nextLevel, stopGame Function

Ảnh có chứa văn bản

Mô tả được tạo tự động

The three functions, let the game start, turn to the next level, and end.

1. Move Shape Algorithm

**Move shapes**:

* Implement KeyListener
* Override keyPressed, create setDeltaX variable so that we can control the shapes moving to the left or the right
* Update method to make the shape just move in the frame
* KeyReleased to make the shapes move faster, so we create the normalspeed and speeddown to manage the moving speed.

1. **RESULT – LIMITATION**
2. **Result**

Our project has followed completely the principle of OOP and the rule of Tetris. Classes and objects in the system are connected relatively and logically. We also success in manipulating the display to conect the user with program. This game can be control by mouse and keyboard.

Morever, **some new features** out of the basic rule are added into our game such as the sound while playing, some button to pause, stop, refresh in order to make the user interface more attractive.

1. **Limitation**

Besides the success of building this game, our group’s project still has some limitations:

* There are too many functions in Board class, which violates the Open/Closed Principle.
* Blocks cannot be rotated when the board is nearly full.
* Does not have the username input.
* Dose not apply design patterns.

1. **CONCLUSION**

Through this project, we had a chance to study and understand deeper the application of Object – Oriented Programming in building a program. Our Tetris game has used the inheritance, the ability to read and write using BufferedWriter and BufferedReader, and other characteristics of Java. We also show the systematical and logical relationships between the classes. However, due to the short time of the course, we cannot finish it as its best. It can be assured that our project will be completed in the future.