Střední průmyslová škola elektrotechnická, Ječná 30

Informační technogie

Ječná 30, 120 00, Praha 2

Build Your Village

Alžběta Olmerová
Informační technologie
2025

Content

1. Goal of the project

The goal of this project was to create a peaceful RPG game. The main objective is to build your own little village. The player should be able to cut down trees, mine ores, and trade with villagers in order to build houses. The game is designed to be easy to understand and suitable for relaxation after work.

2. Description of the game

2.1. Storyline

The player spawns in a small world alongside an old lady and a few homeless villagers. They’re in need of help — their houses have been destroyed, and they’re relying on the player. Can the player build their new homes and save them?

2.2. Characters

2.2.1 Villagers

- **Smith:** sells weapons and tools. He looks like a tough guy but he’s the sweetest.
- **Seller:** This guy knows what he’s doing. His prices are a little higher, but He’s the only one who can sell you these things.
- **Builder:** He’s your new best friend. He’s the only one in the village who knows how to build something. He’s going to help you.

2.2.2. Npc

- **Old lady:** She lives with the villagers, so she also lost her house. She likes to talk, but she’s a sweet old lady. If you need anything just go to her.

2.3 Game mechanics

- Walk around a map (Controls: W,S,A,D)
- Pick up objects
- Talk to Characters
- Trade with villagers
- Destroy objects, such as ores or trees (Controls: space)
- Upgrading weapons (Player needs villagers for that)
- Go to inventory (Controls: E)
- Adjust music and sound effects volume
- (For developers): Info about map and players position (Controls: T)

3. System requirements

The game was developed in Java, specifically in Java 23, so to run the game, the user must have the correct version of Java installed (JDK - Java Development Kit). Besides correct JDK user doesn't need any other external libraries. Program can be run in command line, or in any development enviroment supporting Java, such as *IntelliJ IDEA*, *Eclipse* or *NetBeans*

4. Structure of a program

The project is object-oriented and has a few main classes, such as *Entity, Gamepanel, UI* or *CheckCollision*. The Main class is important for running the program.

Firstly, I would like to clarify that every 'important' class has a method called **update()**. This method updates, for example, the movement of entities while the program is running. These classes also have a **draw()** method, which checks if an entity is visible on the screen; if so, the method draws its image on the screen.

4.1 Entity

The main class for entities - villagers, NPCs and player. Items also extend Entity, which is why this class is so important. The class contains character atributes, but also item ones. The main methods of this class are:

- **setUpImage()**
- **speak()**

4.1.1 setUpImage()

Params:

- imagename: path to image
- width, height: size of the image (size of a tile, half a tile - items, 2*tile - houses)

This method reads the image and uses the **scaledImage()** method to scale it to the requested size. It may not look like an important method, but in reality, it 'defines' every picture in the game - entities, items, even tiles.

4.1.2 speak()

Turns entities towards the player when they're talking (it looks more natural than the player talking to villagers' backs). Also, sets the current dialogue based on the entity and the current phase of the conversation. Basically, this method manages dialogues, which is an important part of the game.

4.2 Gamepanel

The most important class of this project. This class holds all frequently used classes, screen attributes, contains the main game loop, and holds this project together.

4.2.1 run()

An overridden method. The game is programmed to run at 60 FPS, which is handled in this method. That's basically everything this method does - it also calls **update()** and **repaint()** methods under certain conditions.

4.2.2 update()

If gameState == GameState.PLAYING, the method calls update methods on the player, all entities on a map, and interactive tiles (ores and trees).

4.2.3 paintComponent()

A method inherited from the JComponent class. Draws the title screen or the how-to-play screen; otherwise, it draws tiles (map) and also entities. If the user presses T, the method displays debug information.

4.3 CheckCollision

Checks collisions of objects and entities. For example, the player can't walk over water, so when they approach it, they are blocked because there's a collision. The calculation of solid areas and possible collisions is done in this class.

4.4 Player

This class is also important because it carries all the information about the player. It also holds a lot of methods, for example, **pickUpObject(), interactWithNpc(), attack(), or canStackItem()**.

4.4.1 pickUpObject()

Params:

- Index of the item the player wants to pick up

If the player can pick up the object (= index isn't -1), the method checks if the item is stackable using the *canStackItem()* method. If so, the item is added to the player's inventory (under certain conditions, of course).

4.4.2 canStackItem()

Checks if an item is stackable - if its type is MATERIAL, and if it's already in the inventory. You can, of course, only add item/s if there's room in the inventory.

4.5 UI

User interface class. I designed this class so it has 2 'underclasses' - UITrading and UITitleScreen. All of these classes draw on the screen - popup windows, texts, objects, inventory. Everything. They are important for the game flow.

4.5.1 'Main' UI

The most important method in this class is probably the **draw()** method. Inside it is a switch where methods are called based on gameState (methods for drawing different situations, such as inventory, trading screen, or title screen).
Another important method is **drawMessages()**. This method draws every popup message on the screen. So when a player picks up an object, when they level up, etc.

4.5.2 UITrading

This class manages everything that's being processed while trading with villagers - drawing player's inventory, villager's inventory, selecting actions (selling, buying, go back to the game), buying items, selling them, but also giving items to the builder. This class is complex but simple at once.

4.5.3 UITitleScreen

Draws screens - title screen and then a tutorial on how to play. In the end, it also draws the game over screen. This class is pretty 'simple'.

4.6 utilityTool

This class helped me with effective programming. In here are methods that are frequently used, but they're used in many classes. This class is like some kind of toolbox for me.

4.6.1 scaledImage()

Params:

- imageToScale: image the method will be working with
- width, height: required size of the image

This method is used every single time when loading images. Every image is scaled with this method.

4.6.2 drawPopupWindow()

This one is very important in UI classes. This class draws every window. Every pink window is drawn with this class. Yes, there's a lot of them.

4.6.3 goToPlayState()

When the player presses a key and wants to go back to the game, this method is called. It sets *GameState* back to PLAYING, so the player can enjoy the game again.

5 Testing

The best way to test this game is to play it. Mine ores, try to fill your inventory, sell the wrong items, try to escape from the map. Try everything you want; the game should be designed so it catches every possible way of breaking it. If you find any bugs, let me know!

6 User Manual

You don't need to use the mouse. The only time you need it is at the very beginning so the focus is on the game's panel, not anywhere else.

- Movement: W, S, A, D
- Breaking objects: SPACE
- Exit dialogue: ESC
- Open/close inventory: E
- Settings: SHIFT
- Confirm choice: ENTER
- Program details: T

7 Last few words

This project taught me a lot. For me, the main goal of this project was to understand some basic user interface and how games can work. How a keyboard can communicate with the program, how the game loop works. I've learned a lot. One of the biggest problems for me was how to connect everything together so everything is always updated, etc. The YouTube channel RyiSnow helped me a lot. I want to continue with the development of this game. For me personally, this type of game is the best: a chill slow game where you don't need to think and you can just relax.

8 Resources

> *How to Make a 2D Game in Java [@[@RyiSnow]. Online. 2022. Dostupné z: YouTube, https://www.youtube.com/watch?v=om59cwR7psI&list=PL_QPQmz5C6WUF-pOQDsbsKbaBZqXj4qSq. [cit. 2025-05-27].*