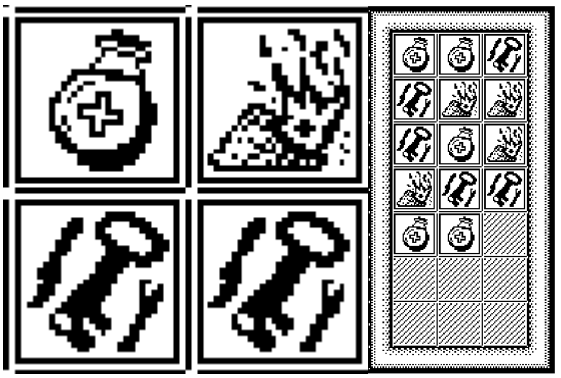
Eldritch Pixus

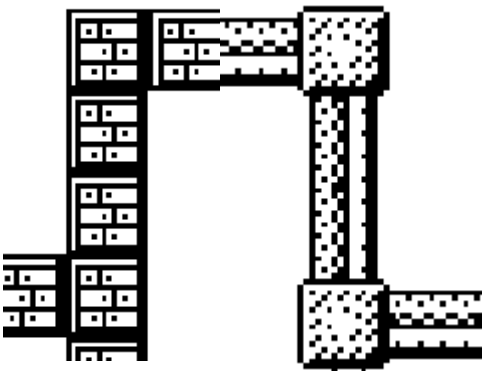
0. Elements:



The inventory:
On the right is the inventory that showcases the player's items.
Left of the there are four items, going from left to right, from top to bottom: potion, flammable powder, key and key.

The game border:
This brick wall marks the limits of the playable area. This is also where the entrance and exit can be found.

The structures:
There's two kinds of structures by default. They do not allow the player to move through them at all. They are either the wooden wall or the stone wall, pictured left and right respectively.



Removable obstacles:
There are three types of obstacles that can and have to be removed: spiders, doors and shrubs.

Dropped items:
Usable items can be found on the ground in the levels. These are the potion, the key and the bag of powder.

The player:
This is the character you control throughout the game.

1. Player's manual:

This is directed at those who wish to play the demo or custom levels and those who simply want to know how the game is supposed to be played.

First of all the **controls** are very simple, as you only need to use the keyboard arrow keys to move around in any of the four cardinal directions.

Secondly the **goal** of the game is to proceed with your character from the starting position to the exit of the current level.

The **levels** consist of a solid, impassable border made out of brick sprites, an entrance in form of a tile resembling a passage, in front of which the player spawns, and an exit in form of a door on the periphery of the playing field, which the player has to reach by trying to move against it.

Furthermore, different levels can have **structures** such as wooden plank walls and stone walls that impede the player's progression.

To unlock the route to beating the level the player will have to deal with **removable obstacles**, such as locked doors (to be distinguished from the level's exit), spiders and shrubs. These can be dealt with by using found keys, potions to strengthen yourself against the giant spiders or bags of flammable powder to help burn down the shrubs at will.

To deal with the aforementioned obstacles there are **items** stored in the **inventory**, which can be either picked up or received at the start of the level. Use them wisely.

Moving against the level's exit will automatically take the player to the next level, if it exists and it's explicitly coded as a follow-up level.

2. Game Master's manual:

As a game master, or level designer, you only need to know a few things.

Mainly, as an editor, you'll work with the playable map, which spans the tiles (1,1) to (22,15) as a rectangle or the columns 1 to 22 and the rows 1 to 15. Tiles use vectors (x,y) for their position. Thereby the top-left tile would have the position (1,1), top-right (22,1), bottom-left (1,15) and bottom-right (22,15).

Columns 0 and 23 and rows 0 and 15 are reserved for the game map's borders.

Before using a level, you'll have to create a map for it. You can do it easily by modifying the Map.jack file and inserting your own map as a method just after the constructor of the class.

You can just copy and use the customMap() method and add upon it. First it is important to specify the locations of the map's entrance, exit and player spawn.

For the exit and entrance choose a non-corner tile on the y-rows 0 or 15 or the x-columns 0 or 23. The player spawn should be within the playable area.

```
////////////////////////////////////////////////
/////////////// CUSTOM MAP BASE ////////////////
////////////////////////////////////////////////

// copy this method and name it to create you own custom map
method Vector customMap() {
    var Vector entrance;
    var Vector exit;
    var Vector vector;
    var int i;

    let exitX = 11;
    let exitY = 0;

    let entrance = Vector.new(11,15);
    let exit = Vector.new(exitX,exitY);

    do Draw.mapAccessPoints(entrance, exit);
    do entrance.dispose();
    do exit.dispose();



    //>>> unique tiles of the map >>>

    //<<< unique tiles of the map <<<

    // return starting position
    let vector = Vector.new(11,14);
    return vector;
}
```

Using the provided template the exit is specified using exitX and exitY, the entrance in "entrance" using a vector (change (11,15) to your own coordinates) and the same for the player spawn just before the return statement (change (11,14) to your own coordinates).

In the "unique tiles of the map" area you can use "*do method();*" to place something on the map.

These are the default methods you can use for populating your map with entities:

placeStoneWall(Vector start, Vector end); (start must be either above or left of end!)
placeWoodenWall(Vector start, Vector end); (start must be either above or left of end!)
placeSpider(Vector position);
placeDoor(Vector position);
placeShrub(Vector position);
placePotion(Vector position);
placeKey(Vector position);
placePowder(Vector position);

Have you built your own map yet? If so, you can put it into the **level rotation**.

The level rotation is defined in the file Game.jack. There you will see multiple default **if** statements. The rules are as following:

1. *the first level is in an if statement of the form "if(level = 0) {...}"*
2. *the next level is the previous level's tag plus 2.*
3. *inside the if statement the levels must contain "do nextLevel();" to prepare for a new level, "let playerPosition = map.<YOUR.MAP>();" to initiate the map.*
4. *Afterwards, inside the if statement, you can use "do inventory.addItem(x)" to add starting items for the player, where x is the code for the item. (0 for potion, 1 for key, 2 for flammable powder by default)*

Now you should be able to compile and run the game starting with the map defined as "level 0" and proceed in steps of 2 through the map "ids".

```
// run the game
while(true) {

    //////////////////////// LEVELS ARE CALLED BELOW ////////////////////////

    // level numbers are always even, level/2 corresponds to the level index
    if(level = 0) {
        // reset to default for the next level
        do nextLevel();
        // select the map
        let playerPosition = map.movementTutorial();
        // add items to inventory
        // <no starting items in this level>

    }
    if(level = 2) {
        do nextLevel();

        let playerPosition = map.itemsTutorial();
    }
    if(level = 4) {
        do nextLevel();

        let playerPosition = map.level1();
    }
    if(level = 6) {
        do nextLevel();

        let playerPosition = map.level2();

        do inventory.addItem(1);
    }
    if(level = 8) {
        do nextLevel();

        let playerPosition = map.win();
        // for pyromaniacs
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
        do inventory.addItem(2);
    }

    //////////////////////// LEVELS ARE CALLED ABOVE ////////////////////////
```

3. Editor’s manual:

If you’re an editor who wants to improve the above code, you should understand that this is still a work in progress and not in a final state, which limits my ability to streamline it as of now.

But things like adding new graphics, structures, tiles, enemies, items should be easily done after understanding the source code. Comments should greatly help.

For questions contact: Marius-Daniel.Petrut@hhu.de

5. Future goals:

As the name implies this game revolves around the theme of eldritch creatures. Things that I would like be in the game but are not yet implemented are among others:

- dynamic enemies and traps
- a bigger variety of tiles and enemies
- a currency and shop system