Solution to GVG-AI 2015 Tutorial 1

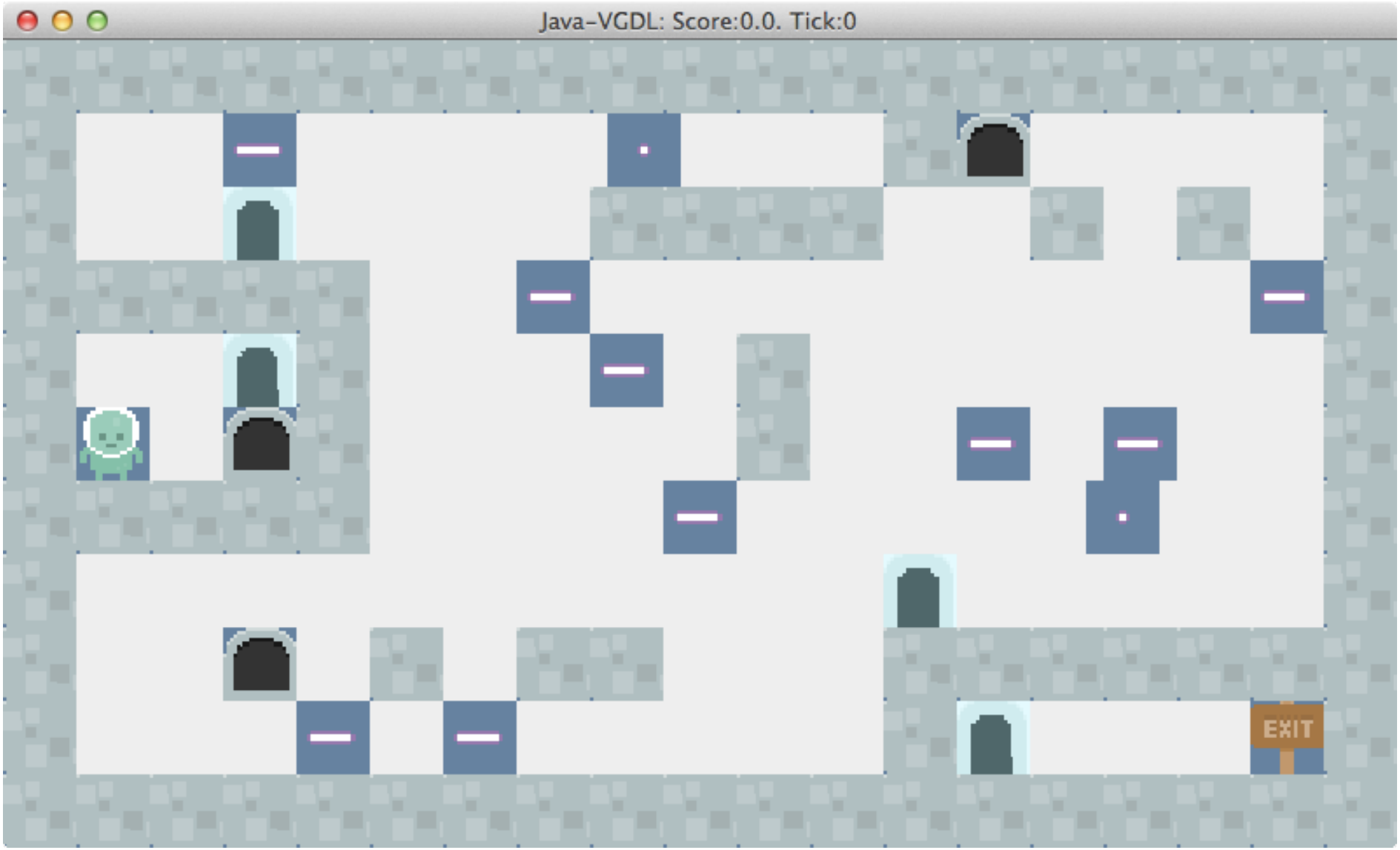
Choosing a Movement

Philipp Kainz

March 26, 2015

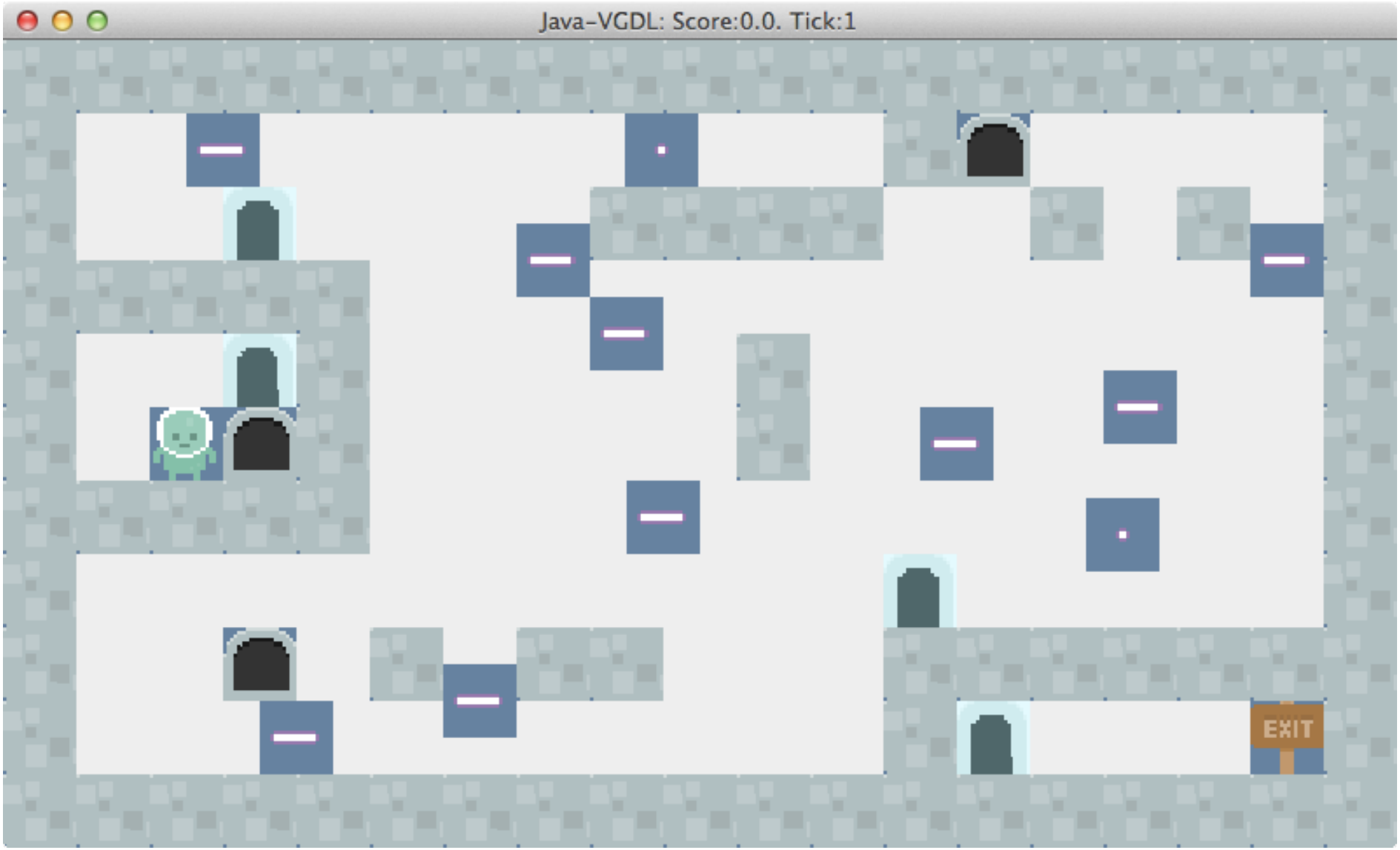
Introducing the Game: PORTALS

Starting position



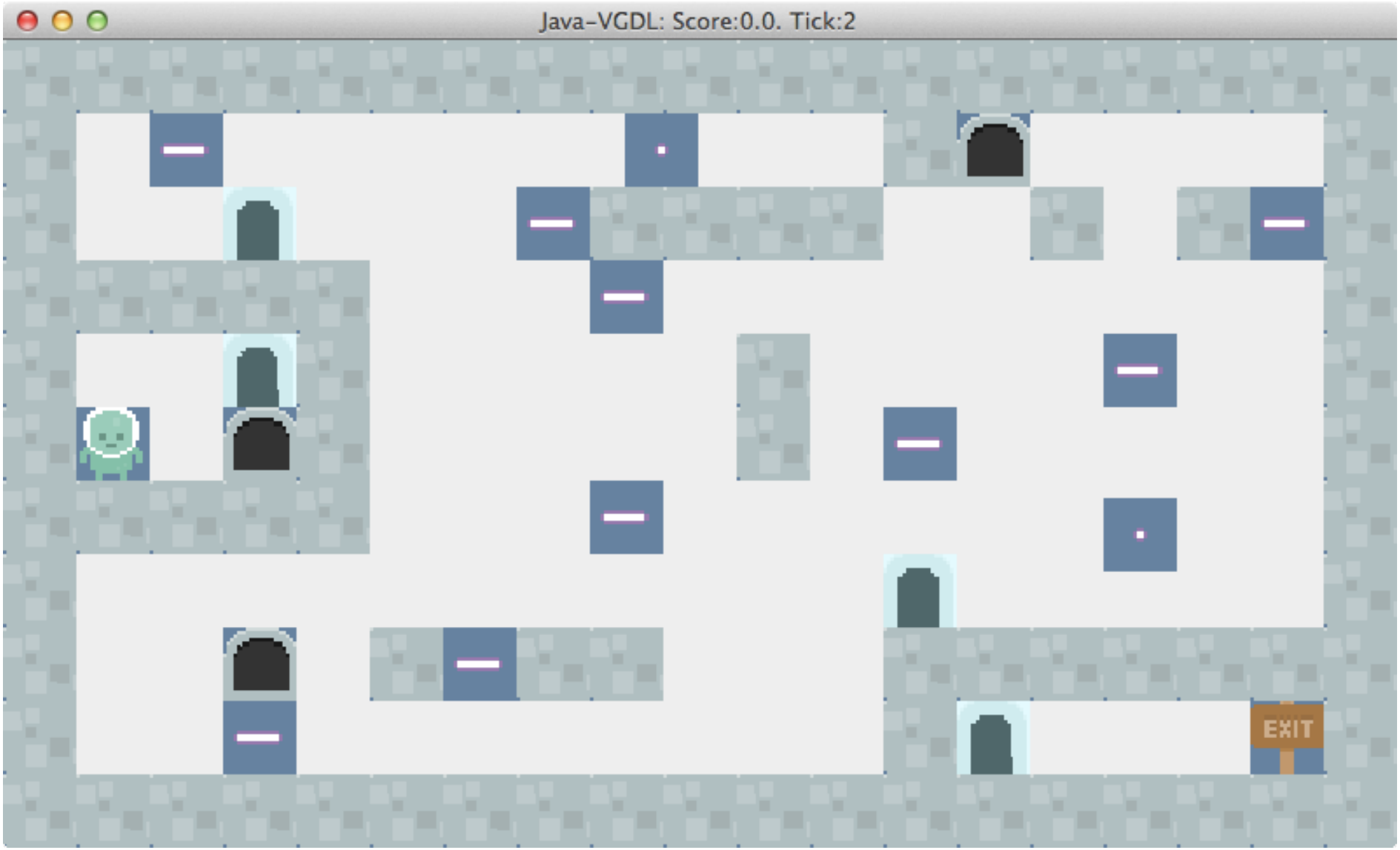
Introducing the Game: PORTALS

Random move 1 goes to the **right**



Introducing the Game: PORTALS

Random move 2 goes to the **left**



Problem: Broken Controller

- Only movements to **left and right** are possible
- The poor guy is trapped in the initial area

```
ArrayList<ACTIONS> available_actions = ↘
    →gameState.getAvailableActions();

ACTIONS action = ACTIONS.ACTION_RIGHT;
int rand_num = randomGenerator.nextInt(2);
if (rand_num == 0){
  action = ACTIONS.ACTION_LEFT;
}
```

Solution

Let the agent choose from a broader set of actions, i.e. from all available actions.

Task: Help the guy escaping the trap

The best approach would be as follows:

1. determine the total number of available actions
2. let the agent choose a random action

How to do that in Java?

Step 1: Get all available actions

Setting up the Eclipse IDE[1] and setting the breakpoint for debugging[2] is explained on our WIKI pages.

```
/*
 * The available actions from the GameStateObservation ↘
     →reference, stored as java.util.ArrayList
 */
ArrayList<ACTIONS> available_actions = ↘
    →gameState.getAvailableActions();

/*
 * Get the size of a dynamic array structure.
 * This should be 4, since:
 *    [ACTION_LEFT, ACTION_RIGHT, ACTION_DOWN, ACTION_UP]
 */
int num_actions = available_actions.size();

/*
 * Prints out to the console.
 */
System.out.println("Number␣of␣actions:␣" + num_actions);
```

[1] https://github.com/benelot/GVG-AI-2015/wiki/Setup-GVG-AI-Development-Environment

[2] https://github.com/benelot/GVG-AI-2015/wiki/Tutorial-1%3A-Choosing-a-Movement

Step 2: Generate a random integer and select movement

- Use the convenient methods from Java's random number generator. `java.util.Random.randInt(x)` returns an integer within [0, x).

```
/*
 * The 'randomGenerator' object has been initialized in the ↘
   →constructor.
 * Remember: num_actions = 4
 */
int random_integer = randomGenerator.nextInt(num_actions);
```

- Access the index (starting at 0 in Java) of the array:

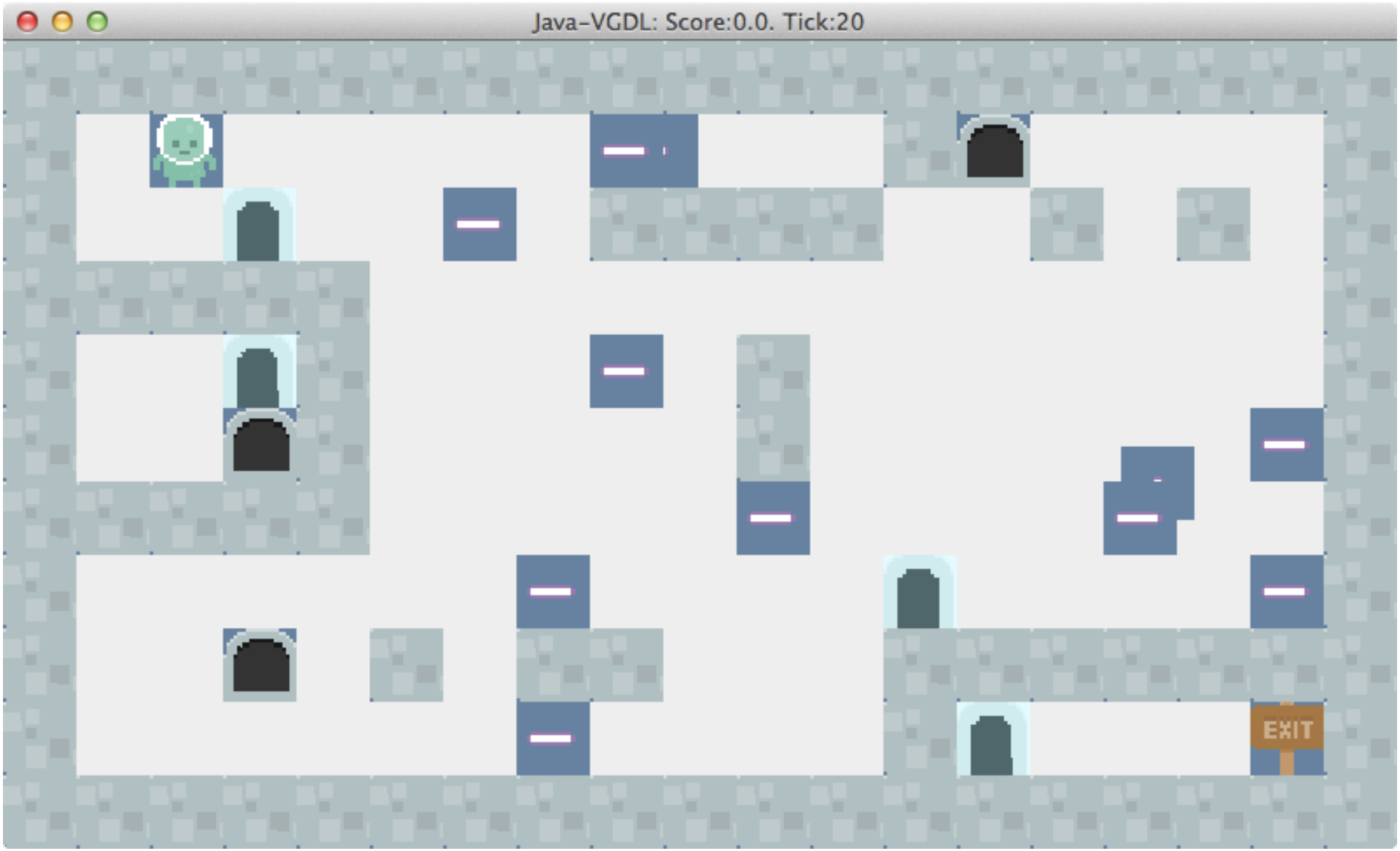
```
ACTION action = available_actions.get(random_integer);
```

Step 3: Putting all together

We can express all previous steps as a single line of code:

```
public ACTIONS act( StateObservation gameState,
                    ElapsedCpuTimer elapsedTimer) {
  ArrayList<ACTIONS> available_actions = ↘
      →gameState.getAvailableActions();
  ACTIONS action = ↘
      →available_actions.get(randomGenerator.nextInt( ↘
      →available_actions.size()));
  return action;
}
```

We freed the guy from his trap!



Looking forward to the next tutorials!