Introduction to Computer Science

JavaScript

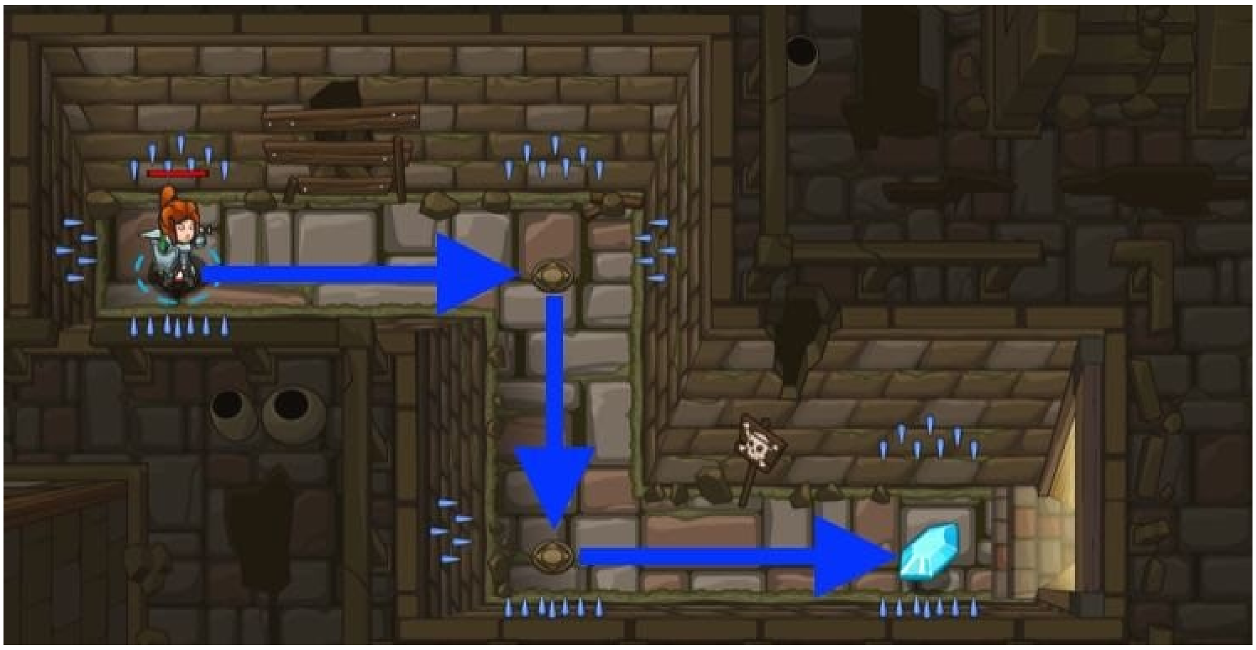
Learn basic syntax, while loops, and the CodeCombat environment.

Table of Contents

#1. Dungeons of Kithgard

Level Overview and Solutions

Intro



Guide your hero by writing a program with code!

Write code in the editor on the right, and click Run when you’re ready. Your hero will read it and follow your instructions.

Move your hero down the hallway without touching the spikes on the walls.

Default Code

```
// Move towards the gem.
// Don’t touch the spikes!
// Type your code below and click Run when you’re done.

hero.moveRight();
```

Overview

The journey begins! To escape the dungeon of Kithgard, your hero has to move. You tell them where to move by writing *code*.

Type a code into the editor to give your hero instructions. Heroes read and execute these instructions themselves when told to. To direct your hero, refer to them in code with `hero`.

Now that you know how to call on your hero, instruct them to move with direction commands, moveDown and moveRight. Write it like this:

```
hero.moveDown();
hero.moveRight();
```

To pass this level, move right, down, and right to grab the gem. Remember, you only need three lines of code.

The code you write here is very similar to the code you might write to tell a computer how to do all kinds of things, from playing music to displaying a web page. You're taking your first steps towards being a coder!

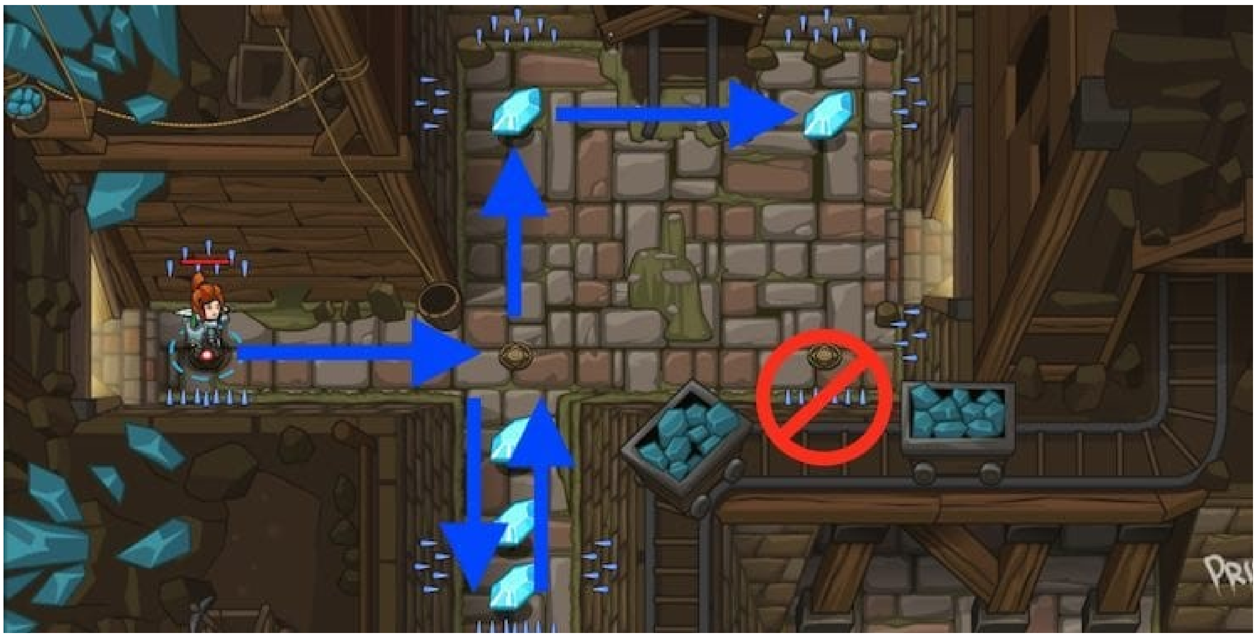
Dungeons of Kithgard Solution

```
// Move towards the gem.
// Don’t touch the spikes!
// Type your code below and click Run when you’re done.

hero.moveRight();
hero.moveDown();
hero.moveRight();
```

#2. Gems in the Deep

Level Overview and Solutions

Intro



Remember how to move, it's important:

```
hero.moveRight()
```

Default Code

```
// Grab all the gems using your movement commands.

hero.moveRight();
```

Overview

Can you remember the lessons from the last level? This will be the same, but you will need to move a lot more. Remember,

```
hero
```

refers to you, the hero.

When you move, you only move as far as the next movement square (look for the small tiles on the ground), so you might have to `moveUp` twice in a row to get to the top of this level from the bottom.

Or you can pass a number as an **argument** to the movement command, to instruct your hero to move more than one space at a time.

For example, you can move up twice by typing:

```
hero.moveUp(2);
```

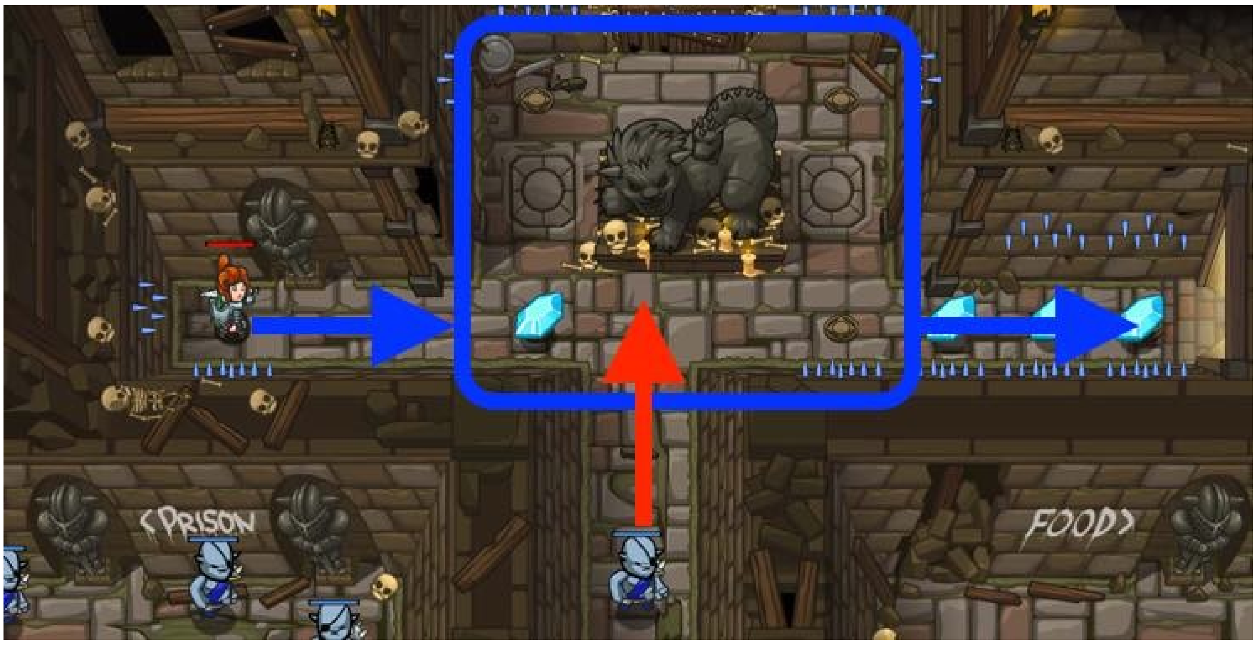
Gems in the Deep Solution

```
// Grab all the gems using your movement commands.

hero.moveRight();
hero.moveDown();
hero.moveUp();
hero.moveUp();
hero.moveRight();
```

#3. Shadow Guard

Level Overview and Solutions

Intro



Use cover to avoid being seen.

Default Code

```
// Avoid being seen by the ogre. Collect all the gems.
hero.moveRight();
```

Overview

This is all about skills of secrecy. You don't have a weapon yet, so you can't fight the ogre munchkin who guards the path.

Instead, try moving up, behind the statue, so he doesn’t see you. Then you can get the gems and run to the finish undetected.

Shadow Guard Solution

```
// Avoid being seen by the ogre. Collect all the gems.
hero.moveRight();
hero.moveUp();
hero.moveRight();
hero.moveDown();
hero.moveRight();
```

#3a. Kounter Kithwise

Level Overview and Solutions

Intro

Avoid the Munchkins by picking your path carefully!

Default Code

```
// Avoid the ogres and grab the gem.
```

Overview

Sometimes it's all about timing. Grab the gem, but move the same way the ogre patrol does, so that they don't see you.

You can drag the playback timeline to see what happened the last time you ran your code.

Kounter Kithwise Solution

```
// Avoid the ogres and grab the gem.

hero.moveDown(2);
hero.moveRight();
hero.moveUp();
hero.moveRight();
```

#3b. Crawlways of Kithgard

Level Overview and Solutions

Intro

Time your movement carefully to avoid the patrol!

Default Code

```
// Avoid being seen by the ogres.
```

Overview

You must use your wits to solve the puzzle. How can you move such that you aren't crossing the hallway at the same time as the ogre munchkins?

Remember, you can always backtrack the way you came.

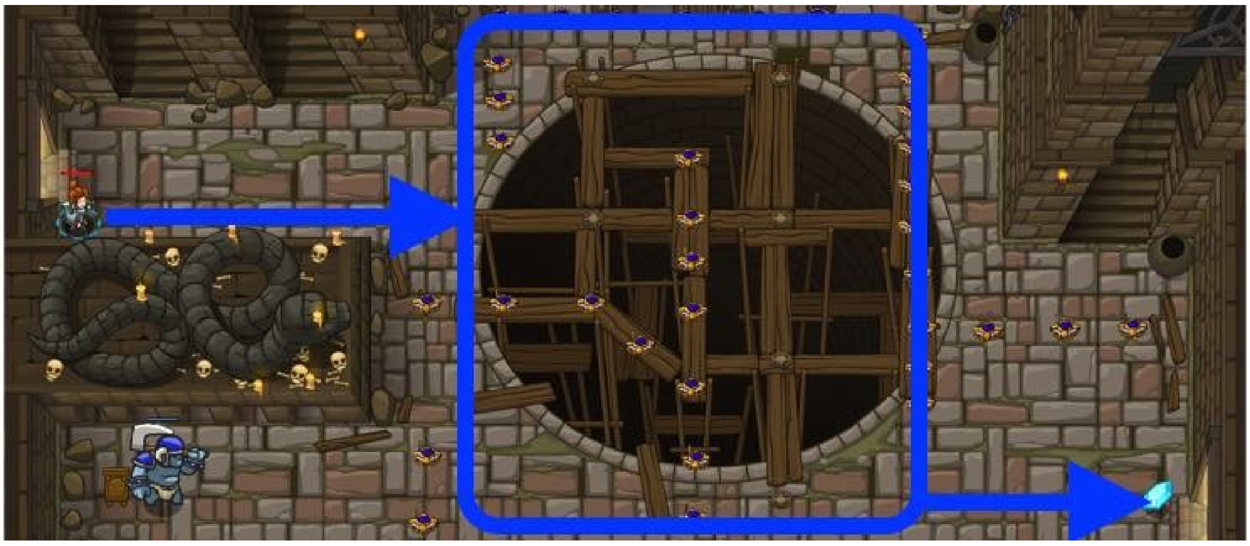
Crawlways of Kithgard Solution

```
// Avoid being seen by the ogres.

hero.moveRight();
hero.moveLeft();
hero.moveRight(2);
```

#4. Enemy Mine

Level Overview and Solutions

Intro



You can use arguments to optimize your code. Instead of:

```
hero.moveRight();
hero.moveRight();
```

You can use:

```
hero.moveRight(2);
```

Default Code

```
// Use arguments with move statements to move farther.
hero.moveRight(3);
```

Overview

The floor is littered with Fire Traps, but there's a safe path through to the gem.

When you call a method like `moveRight()` you can sometimes give extra information to the method to modify what it does. This extra information is referred to as "arguments" or "parameters".

You can pass an argument to the `moveRight()` method like this: `moveRight(3)` . This tells moveRight() to make your hero move 3 spaces to the right instead of 1.

Enemy Mine Solution

```
// Use arguments with move statements to move farther.
hero.moveRight(3);
hero.moveUp();
hero.moveRight();
hero.moveDown(3);
hero.moveRight(2);
```

#4a. Illusory Interruption

Level Overview and Solutions

Intro

Step on the **Red X** to launch the decoy.

Move to the gem and escape!

Default Code

```
// Use the Decoy to distract the guards by stepping on the X.
```

Overview

You can't sneak past the guards unless they're distracted. Luckily, someone left a decoy nearby.

Stepping on the X will activate the decoy.

Tip: You can move multiple spaces by passing arguments to the move command, like `moveRight(3)`.

Illusory Interruption Solution

```
// Use the Decoy to distract the guards by stepping on the X.

hero.moveRight();
hero.moveDown(2);
hero.moveUp(2);
hero.moveRight(3);
```

#4b. Forgetful Gemsmith

Level Overview and Solutions

Intro

Collect the gems by using your move commands!

```
hero.moveRight();
hero.moveDown();
```

Default Code

```
// Grab the gems and go to the exit.
```

Overview

This one takes several commands, so make sure to use the autocomplete to write your code faster. You can type `r`, Enter to autocomplete a `moveRight()` command.

After you're done practicing simple movement with this level, it's time to learn `attack`ing, so keep trying!

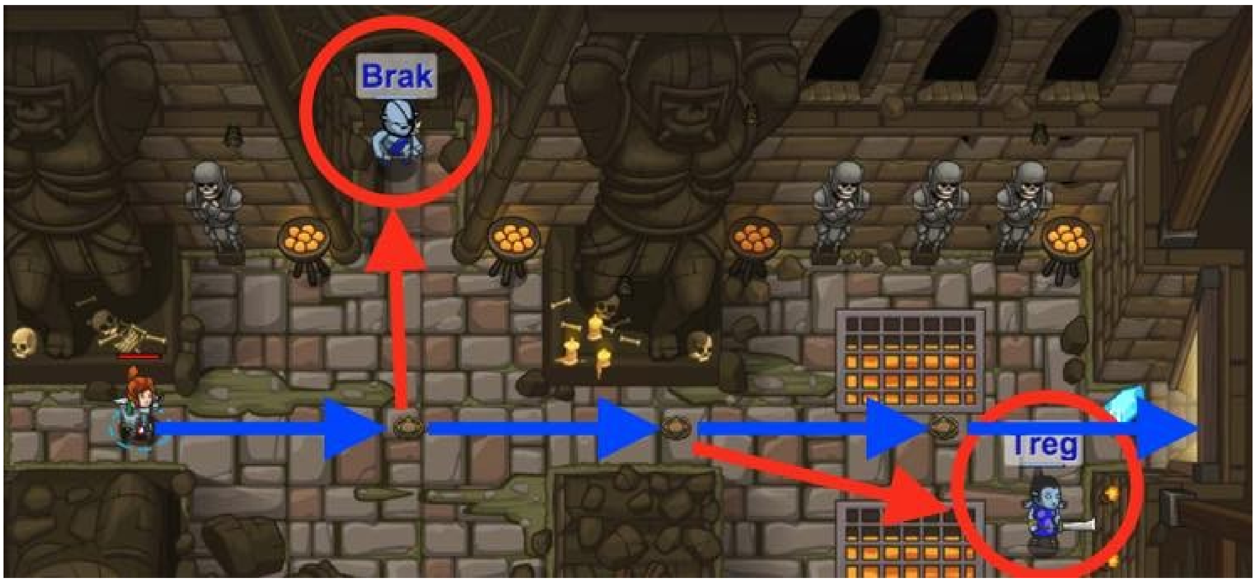
Forgetful Gemsmith Solution

```
// Grab the gems and go to the exit.

hero.moveRight();
hero.moveDown();
hero.moveRight(2);
hero.moveUp();
hero.moveRight();
```

#5. True Names

Level Overview and Solutions

Intro



Be sure to attack each ogre twice.

Use the `attack` method to attack an enemy by their `"Name"`.

Capitalization is important!

Default Code

```
// Defend against Brak and Treg!
// You must attack small ogres twice.

hero.moveRight();
hero.attack("Brak");
hero.attack("Brak");
```

Overview

Keep in mind a few things to beat this level:

1. You need to attack each ogre munchkin **twice** to defeat it.
2. Spell the names properly, with capitalization! `"Brak"` and `"Treg"`.
3. Put the names in quotes to make them into strings. Strings are a type of programming data. They represent text.
4. After you defeat `"Brak"` you should `moveRight()` to get the gem.

5. Then, defeat `"Treg"` by attacking them twice.
6. It's no problem if you die; you can always keep trying.

True Names Solution

```
// Defend against Brak and Treg!
// You must attack small ogres twice.

hero.moveRight();
hero.attack("Brak");
hero.attack("Brak");
hero.moveRight();
hero.attack("Treg");
hero.attack("Treg");
```

#5a. Favorable Odds

Level Overview and Solutions

Intro

Coming soon!

Default Code

```
// Attack both ogres and grab the gem.
hero.moveRight();
hero.attack("Krug");
hero.attack("Krug");
```

Overview

Remember to hit ogre munchkins twice, and to put their names capitalized in quotes: "Krug" and "Grump".

If you buy a powerful sword, you can take down munchkins in one hit.

Favorable Odds Solution

```
// Attack both ogres and grab the gem.
hero.moveRight();
hero.attack("Krug");
hero.attack("Krug");
hero.moveRight();
hero.moveUp();
hero.attack("Grump");
hero.attack("Grump");
hero.moveLeft();
hero.moveLeft();
```

#5b. The Raised Sword

Level Overview and Solutions

Intro

Attack each ogre by name. Remember, each ogre takes two hits!

Default Code

```
// Defeat the ogres.
// Remember that they each take two hits.
```

Overview

Make sure to have enough armor to win the fight! You should buy the Tarnished Bronze Breastplate in the item shop.

Remember, each munchkin takes two hits to defeat. Try to defeat them in the order they come at you so that they don't land extra hits on you.

The Raised Sword Solution

```
// Defeat the ogres.
// Remember that they each take two hits.

hero.attack("Rig");
hero.attack("Rig");

hero.attack("Gurt");
hero.attack("Gurt");

hero.attack("Ack");
hero.attack("Ack");
```

#6. Cell Commentary

Level Overview and Solutions

Intro

Comments are a way for a programmer to explain their code to another programmer.

In CodeCombat, comments are helpful hints on what to write and how to structure your code!

```
// JavaScript comments start with //. Read them for instructions!
```

Default Code

```
hero.say("What's the password?");
// Use the "say()" function to say the password.
// The password is: "Achoo"
```

Overview

Comments

Comments are a way for two programmers to communicate with each other. They are even useful for a single programmer to remember what they were doing to resume later!

In CodeCombat, we add comments to help you structure your code and give vital hints. Read them to understand what the objective is and what code you need to write to accomplish it.

In this level you will need to read the comments we provide to find the answer, to escape the prison cell!

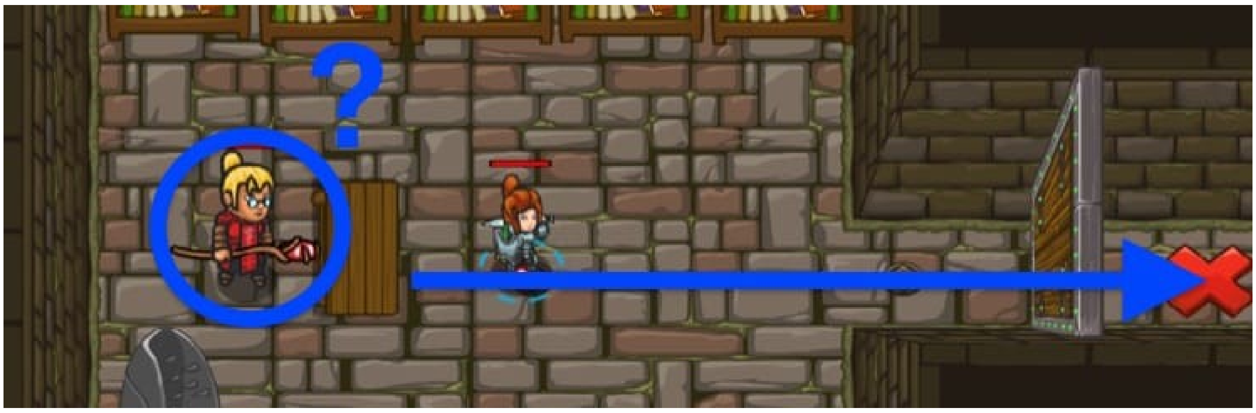
Cell Commentary Solution

```
hero.say("What's the password?");
// Use the "say()" function to say the password.
// The password is: "Achoo"
hero.say("Achoo");
hero.moveUp();
hero.moveUp();
```

#7. Kithgard Librarian

Level Overview and Solutions

Intro



Most levels have Hints that can help you when you're stuck.

Click 'Next' to scroll through all the Hints for a level.

For this level, click 'Next' to find the password for the library door!

Default Code

```
// You need to say a password to open the Library Door!
// The password is in the Hints!
// Click the blue Hints button above the code window.
// If you ever get stuck, click Hints!

hero.moveRight();
hero.say("I don't know the password!");  // ∆
```

Overview

Welcome to Hints!

Most levels have hints that contain extra information to help you complete your goals and learn new concepts.

Like this:

The password for the Library Door is `"Hush"`.

Tip: Use the `say` command to say the password!

Tip: Remember to use quotes around the password to make it a string, and use correct capitalization.

Be sure to check Hints on future levels if you're having trouble!

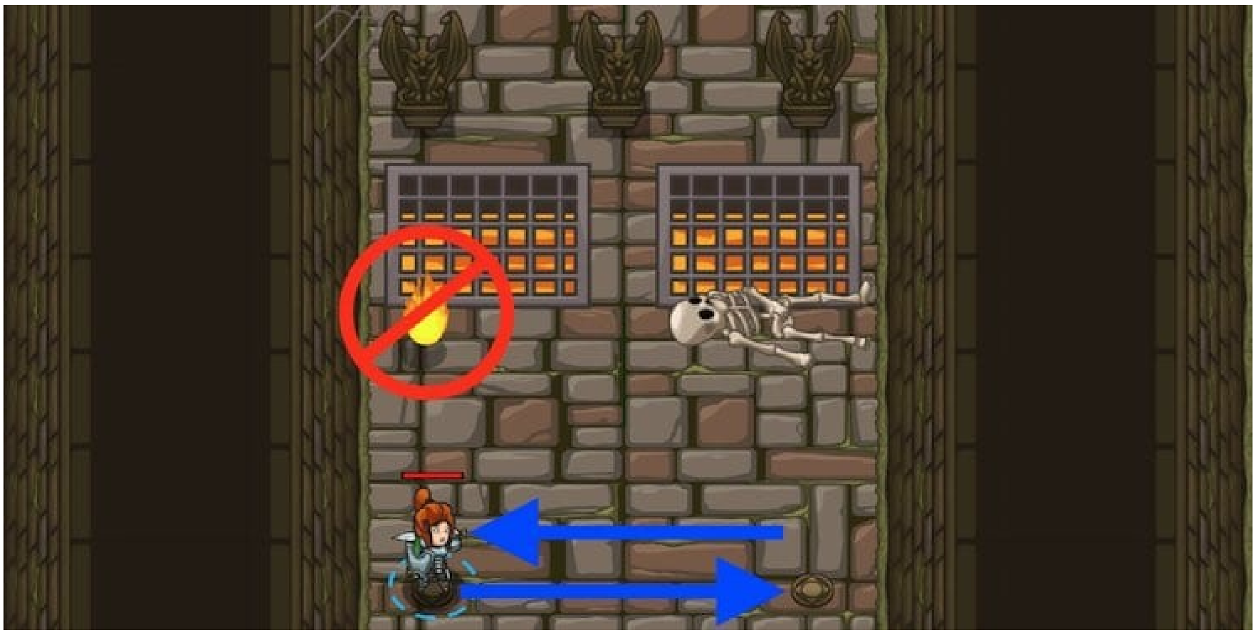
Kithgard Librarian Solution

```
// You need to say a password to open the Library Door!
// The password is in the Hints!
// Click the blue Hints button above the code window.
// If you ever get stuck, click Hints!

hero.moveRight();
hero.say("Hush");
hero.moveRight();
```

#8. Fire Dancing

Level Overview and Solutions

Intro



Avoid the fireballs by dancing **right** and **left**.

Use a **while-true loop** to repeat forever like this:

```
while (true) {
    hero.moveLeft(); // This will happen over and over.
}
```

Default Code

```
// Code normally executes in the order it's written.
// Loops repeat a block of code multiple times.

while(true) {
    hero.moveRight();
    // Add a moveLeft command to the loop here

}
```

Overview

Code normally execudes in the order it's written. **Loops** allow you to repeat a block of code multiple times without having to re-type it.

How To Use while-true Loops

First, we start a loop with the `while` keyword. This tells your program **WHILE** something is true, repeat the **body** of the loop.

For now we want our loops to continue forever, so we'll use a **while-true loop**. Because true is always true!

Don't worry about this true stuff too much for now. We'll explain it more later. Just remember that a **while-true loop** is a loop that never ends.

This is how you code a **while-true loop**:

```
// Start the while-true loop with "while(true) {"
// Anything between the { and } is considered INSIDE the loop.
while(true) {
    hero.moveRight();
    hero.moveLeft();
}

hero.say("This line is not inside the loop!");

// Tip: the indentation (spaces at the start of the lines) is optional, but makes your code easier
    to read!
```

Tip: You can put whatever you want inside a while-true loop! But for this level, we only need to repeat two commands: `moveRight()` and `moveLeft()`!

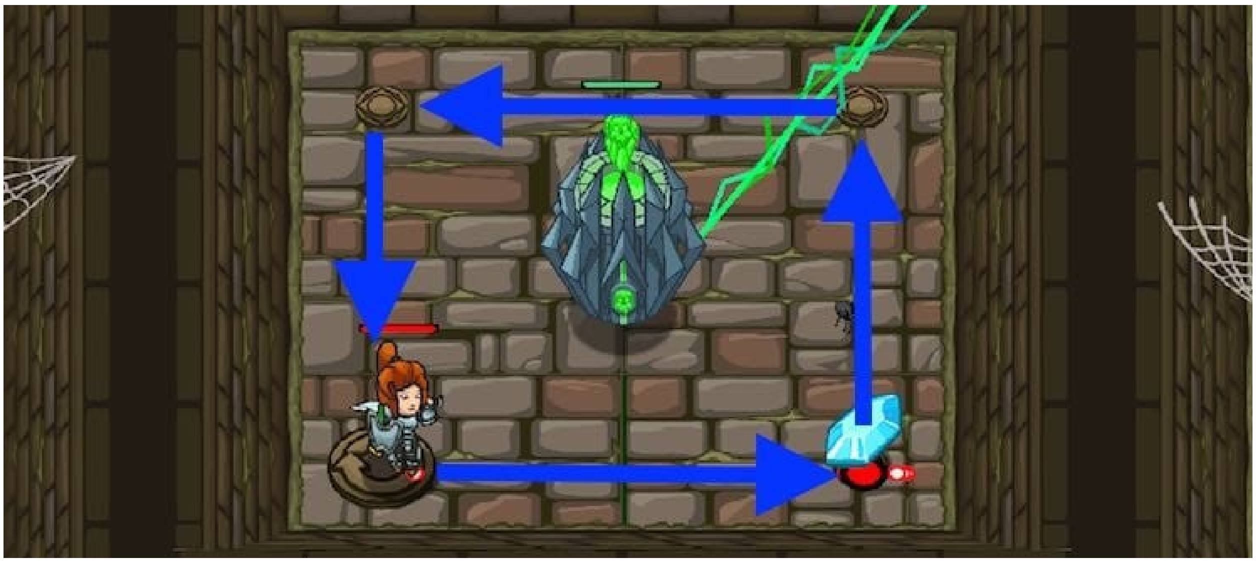
Fire Dancing Solution

```
// Code normally executes in the order it's written.
// Loops repeat a block of code multiple times.

while(true) {
    hero.moveRight();
    // Add a moveLeft command to the loop here
    hero.moveLeft();
}
```

#9. Loop Da Loop

Level Overview and Solutions

Intro



You only need *one* **while-true loop** containing **4 commands** to survive this level!

Double check your indentation!

Default Code

```
// Code inside of a while-true loop will repeat forever!

while(true) {
    // Move right
    hero.moveRight();
    // Move up

    // Move left

    // Move down

}
```

Overview

You can survive this level using one **while-true loop** containing just **4 commands**!

Make sure the commands you add are **inside** the **while-true loop**. Double check your indentation!

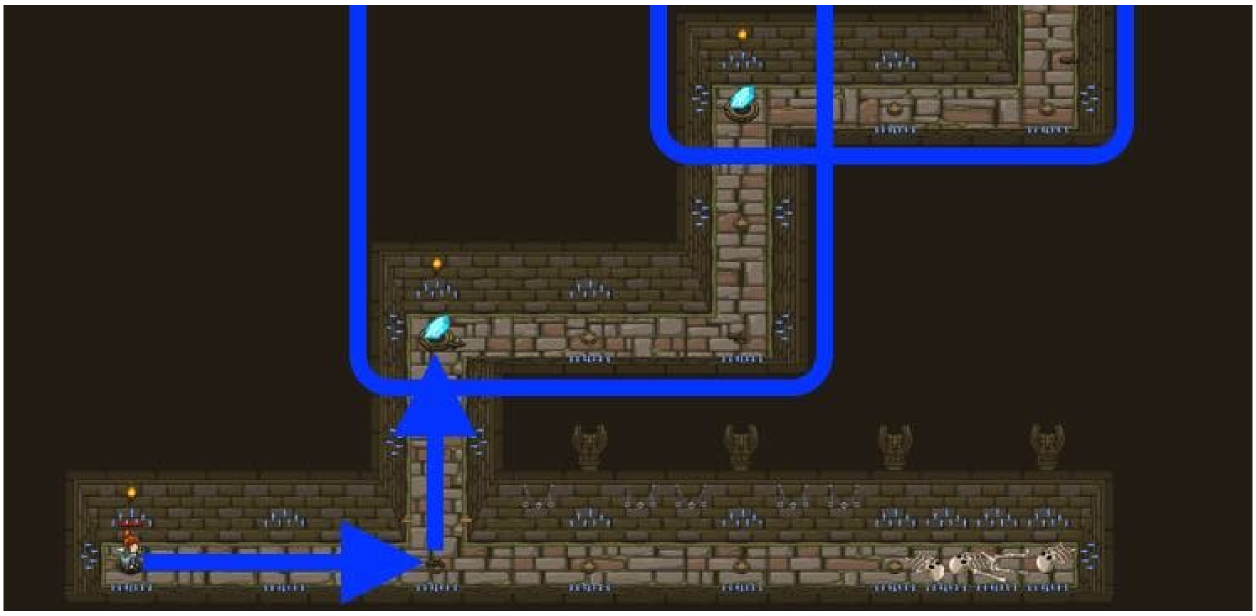
Loop Da Loop Solution

```
// Code inside of a while-true loop will repeat forever!

while(true) {
    // Move right
    hero.moveRight();
    // Move up
    hero.moveUp();
    // Move left
    hero.moveLeft();
    // Move down
    hero.moveDown();
}
```

#10. Haunted Kithmaze

Level Overview and Solutions

Intro



Double check your indentation!

You only need **one while-true loop** containing **four** commands in this level.

Default Code

```
// Loops are a better way of doing repetitive things.

while(true) {
    // Add commands in here to repeat.
    hero.moveRight();
    hero.moveRight();

}
```

Overview

Loops let you repeat the same code over and over. You can do this level in just four commands with a **while-true loop**.

Tip: the hallway needs **two movements to the right**, and then **two movements up**. From there, you can just let the **while-true loop** repeat to do the rest.

Make sure that your movement commands are **inside the loop** so that they repeat!

Haunted Kithmaze Solution

```
// Loops are a better way of doing repetitive things.

while(true) {
    // Add commands in here to repeat.
    hero.moveRight();
    hero.moveRight();
    hero.moveUp();
    hero.moveUp();
}
```

#10a. Descending Further

Level Overview and Solutions

Intro

Coming soon!

Default Code

```
// Navigate the maze in less than 5 statements.
```

Overview

For the bonus, use a loop to do this level in only four code statements. (Empty lines and comments don't count, and multiple statements on the same line do count.)

Hover over the loop documentation in the bottom right to see examples of the loop syntax.

Descending Further Solution

```
// Navigate the maze in less than 5 statements.

while(true) {
    hero.moveRight(2);
    hero.moveDown();
}
```

#10b. Riddling Kithmaze

Level Overview and Solutions

Intro

Coming soon!

Default Code

```
// Loops are a better way of doing repetitive things.

while(true) {
    // Add commands in here to repeat.
    hero.moveRight();
    hero.moveDown();

}
```

Overview

Count carefully how many movements you need inside your loop to solve the maze, and where you need to repeat your loop. Remember, you should only use one loop per level, and make sure all your code is inside the loop.

Hover over the loop documentation to see an example of the loop syntax.

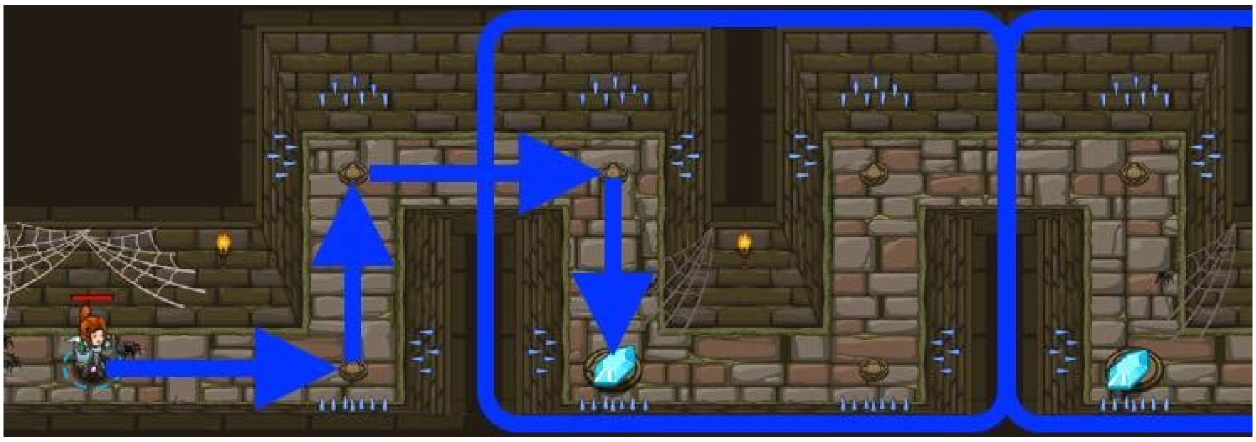
Riddling Kithmaze Solution

```
// Loops are a better way of doing repetitive things.

while(true) {
    // Add commands in here to repeat.
    hero.moveRight();
    hero.moveDown();
    hero.moveRight(2);
    hero.moveUp();
}
```

#11. The Second Kithmaze

Level Overview and Solutions

Intro



Remember, you only need **one while-true loop**. It will repeat!

Default Code

```
// Finish the while-true loop to navigate the maze!

while(true) {
    hero.moveRight();
    // Type in 3 more move commands to finish the maze:

}
```

Overview

Carefully count how many movements you need inside your **while-true loop** to solve the maze!

Remember, you should only use one **while-true loop** per level, and make sure all your code is inside the loop.

Hover over the **while-true loop** documentation in the lower right to see an example.

The Second Kithmaze Solution

```
// Finish the while-true loop to navigate the maze!

while (true) {
    hero.moveRight();
    // Type in 3 more move commands to finish the maze:
    hero.moveUp();
    hero.moveRight();
    hero.moveDown();
}
```

#11a. Radiant Aura

Level Overview and Solutions

Intro

Coming soon!

Default Code

```
// Pick up a lightstone to make skeletons flee from you for a short time.
```

Overview

Skeletons are too tough to fight. The Lightstones can keep them away from you, but only for a little while!

All you need to do is:

- Grab a Lightstone
- Move past a skeleton
- Repeat

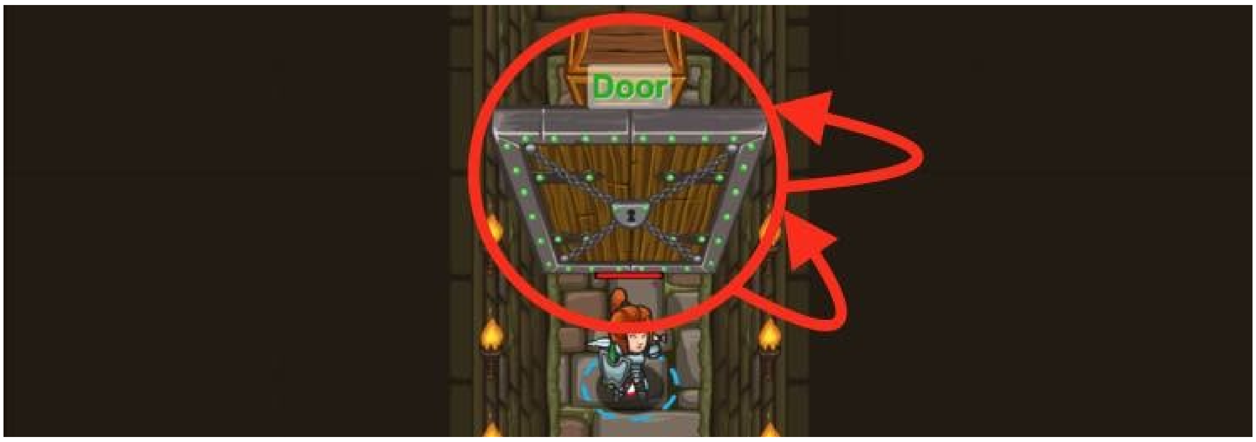
Radiant Aura Solution

```
// Pick up a lightstone to make skeletons flee from you for a short time.

while(true) {
    hero.moveUp();
    hero.moveDown();
    hero.moveRight(2);
}
```

#12. Dread Door

Level Overview and Solutions

Intro



You can use **while-true loops** with any method like:

```
while(true) {
    hero.moveRight();
    hero.attack("Brak");
}
```

Default Code

```
// Attack the door!
// It will take many hits, so use a "while-true" loop.
```

Overview

You can combine **while-true loops** and `attack` to easily destroy things that take more than one hit. Like this door.

```
while(true) {
    hero.attack("Door");
}
```

You can attack the door by its name, which is `"Door"` .

With looping and attacking, you can do this level in just two lines of code.

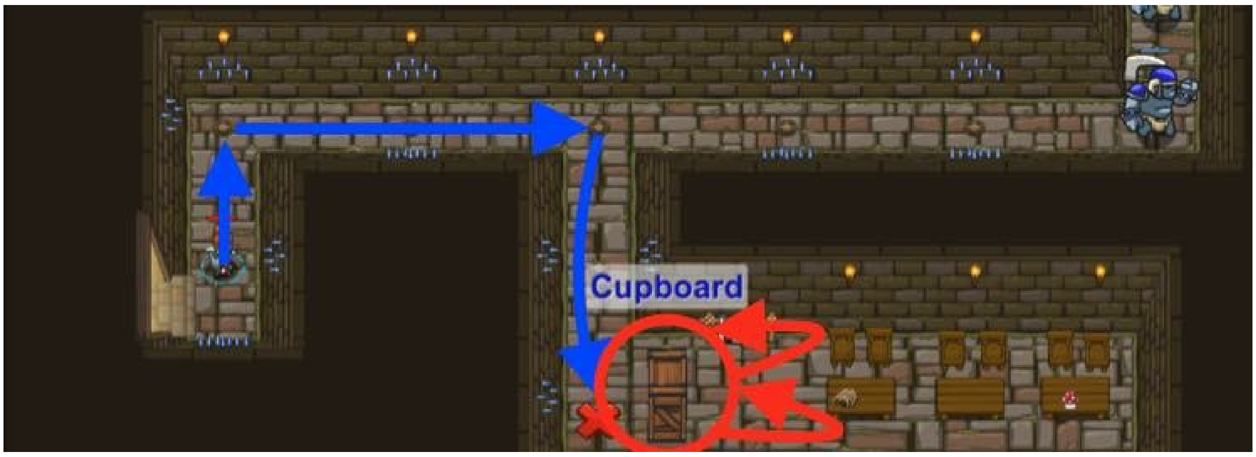
Dread Door Solution

```
// Attack the door!
// It will take many hits, so use a "while-true" loop.

while (true) {
    hero.attack("Door");
}
```

#13. Cupboards of Kithgard

Level Overview and Solutions

Intro



You can perform any action before a `while-true loop`.

```
hero.moveUp();
while(true) {
    hero.attack("Brak");
}
```

Default Code

```
// There may be something around to help you!

// First, move to the Cupboard.

// Then, attack the "Cupboard" inside a while-true loop.
```

Overview

The ogre guards might be too much for you to handle. Maybe you'll find something useful in the `"Cupboard"`?

First, move close to the `"Cupboard"` (stand on the red X). It looks locked, so you'll have to attack it repeatedly using a **while-true loop** to break it open.

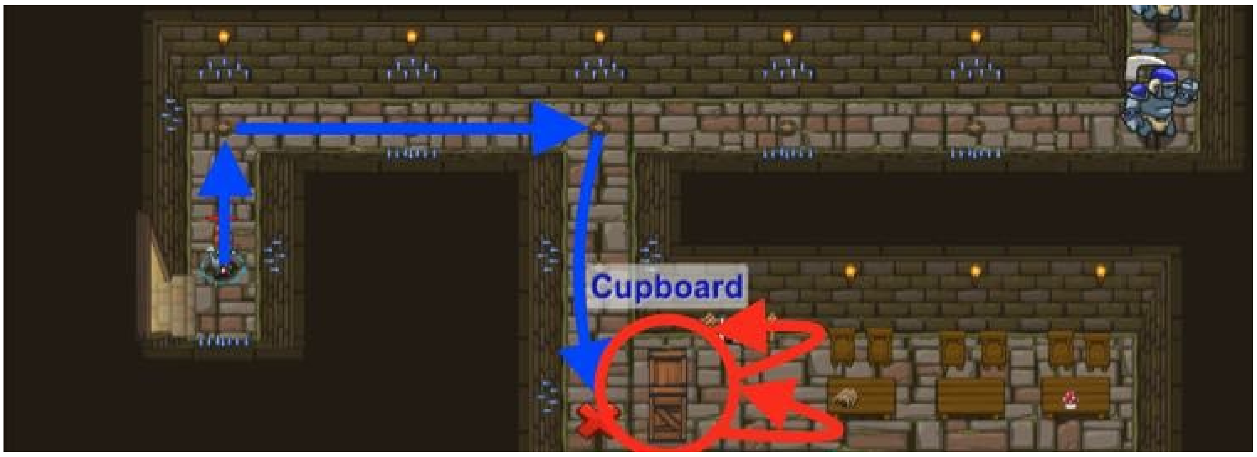
Cupboards of Kithgard Solution

```
// There may be something around to help you!

// First, move to the Cupboard.
hero.moveUp();
hero.moveRight(2);
hero.moveDown(2);

// Then, attack the "Cupboard" inside a while-true loop.
while (true) {
    hero.attack("Cupboard");
}
```

#13a. Cupboards of Kithgard A

Level Overview and Solutions

Intro



You can perform any action before a `while-true loop`.

```
hero.moveUp();
while(true) {
    hero.attack("Brak");
}
```

Default Code

```
// There may be something around to help you!

// First, move to the Cupboard.

// Then, attack the "Cupboard" inside a while-true loop.
```

Overview

The ogre guards might be too much for you to handle. Maybe you'll find something useful in the `"Cupboard"`?

First, move close to the `"Cupboard"` (stand on the red X). It looks locked, so you'll have to attack it repeatedly using a **while-true loop** to break it open.

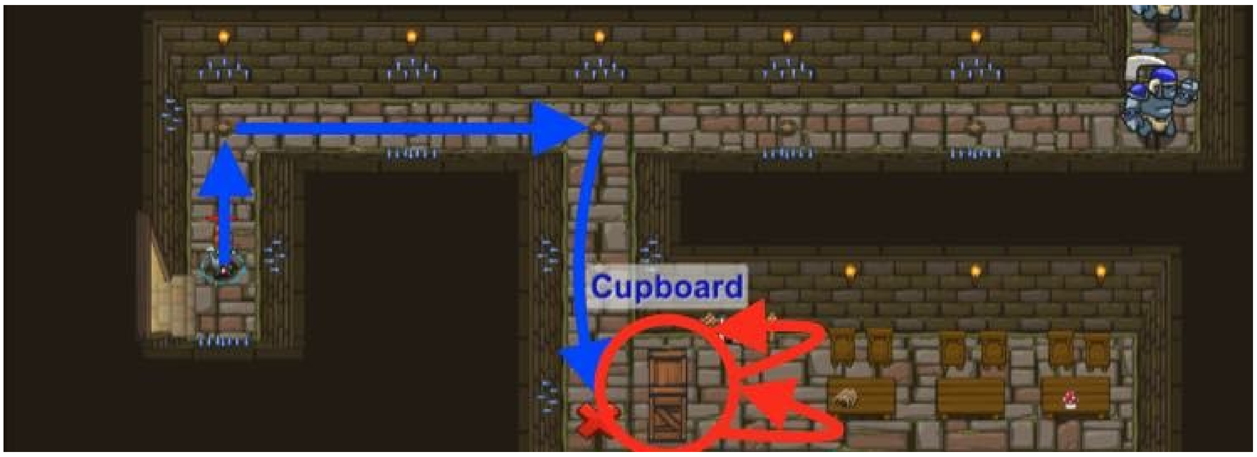
Cupboards of Kithgard A Solution

```
// There may be something around to help you!

// First, move to the Cupboard.
hero.moveDown();
hero.moveLeft(2);
hero.moveUp(2);

// Then, attack the "Cupboard" inside a while-true loop.
while (true) {
    hero.attack("Cupboard");
}
```

#13b. Cupboards of Kithgard B

Level Overview and Solutions

Intro



You can perform any action before a `while-true loop`.

```
hero.moveUp();
while(true) {
    hero.attack("Brak");
}
```

Default Code

```
// There may be something around to help you!

// First, move to the Cupboard.

// Then, attack the "Cupboard" inside a while-true loop.
```

Overview

The ogre guards might be too much for you to handle. Maybe you'll find something useful in the `"Cupboard"`?

First, move close to the `"Cupboard"` (stand on the red X). It looks locked, so you'll have to attack it repeatedly using a **while-true loop** to break it open.

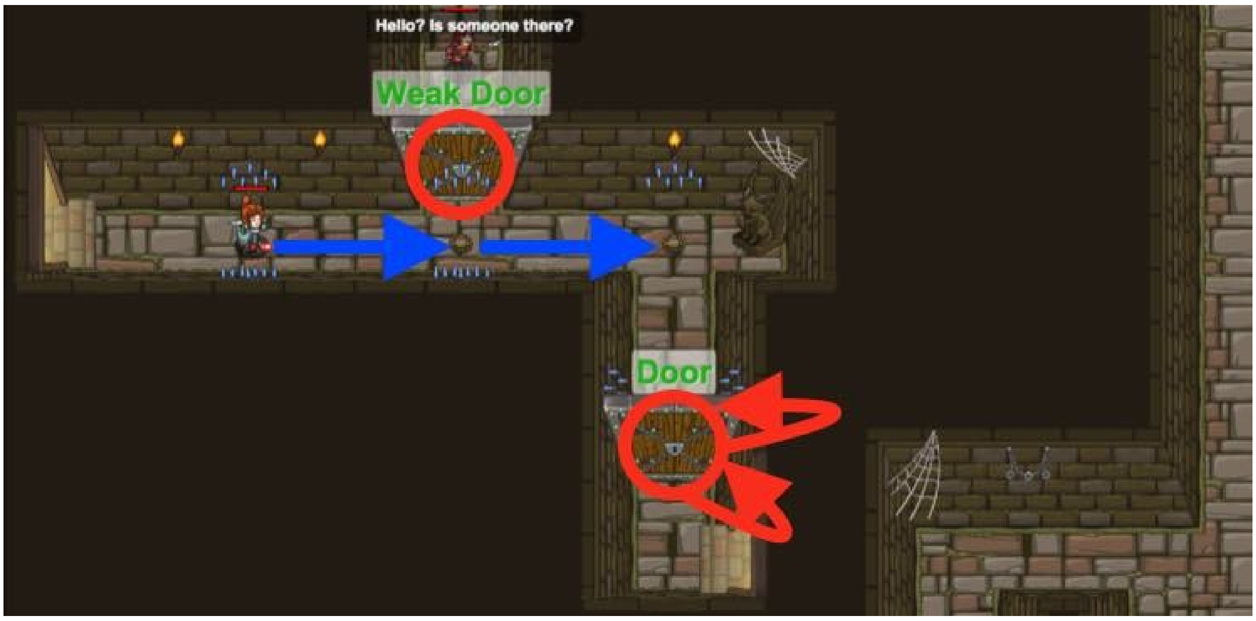
Cupboards of Kithgard B Solution

```
// There may be something around to help you!

// First, move to the Cupboard.
hero.moveRight();
hero.moveDown();
hero.moveRight();
hero.moveDown(2);

// Then, attack the "Cupboard" inside a while-true loop.
while (true) {
    hero.attack("Cupboard");
}
```

#14. Breakout

Level Overview and Solutions

Intro



Free your friend by attacking the `"Weak Door"` so you have more time to break the stronger `"Door"` with a **while-true loop**.

Default Code

```
// Free your friend, then clear a path to escape!
```

Overview

You'll need that soldier to protect you, so first attack the `"Weak Door"` to free her.

Then use a **while-true loop** to attack the `"Door"` while your new friend holds off the munchkins.

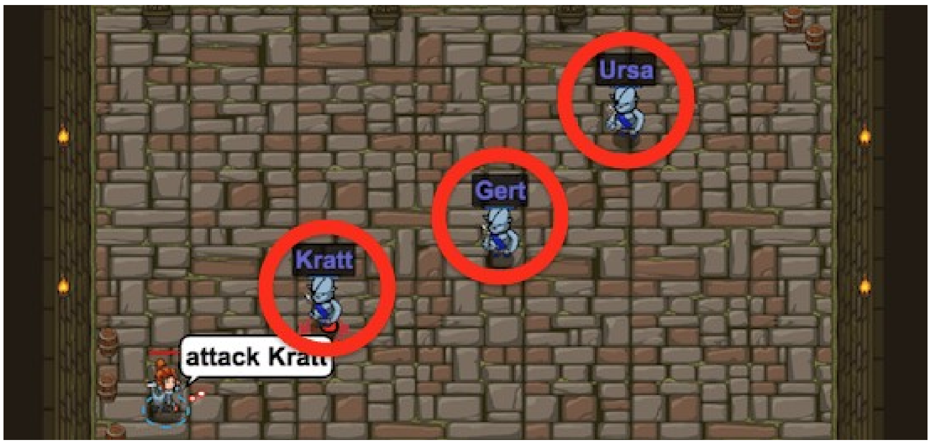
Breakout Solution

```
// Free your friend, then clear a path to escape!

hero.moveRight();
hero.attack("Weak Door");
hero.moveRight();
hero.moveDown();
while (true) {
    hero.attack("Door");
}
```

#15. Known Enemy

Level Overview and Solutions

Intro



Declare a variable like this:

```
var enemy1 = "Kratt";
```

When you use quotes: `"Kratt"` , you are making a **string**.

When you don't use quotes: `enemy1` , you are referencing the `enemy1` **variable**.

Default Code

```
// You can use a variable like a nametag.

var enemy1 = "Kratt";
var enemy2 = "Gert";
var enemy3 = "Ursa";

hero.attack(enemy1);
hero.attack(enemy1);

hero.attack(enemy2);
```

Overview

Up until now, you have been doing three things:

1. Calling **methods** (commands like `moveRight()`)
2. Passing **strings** (quoted pieces of text like "Treg") as arguments to the methods
3. Using **while-true loops** to repeat your methods over and over.

Now you are learning how to use **variables**: symbols that represent data. The variable's value can **vary** as you store new data in it, which is why it's called a variable.

It's a pain to type the names of ogres multiple times, so in this level you use three variables to store the ogre names. Then when you go to attack, you can use the variable (`enemy1`) to represent the string that is stored in it (`"Kratt"`).

Declare variables like so:

```
var enemy1 = "Kratt";
```

When you use quotes: `"Kratt"` , you are making a **string**.

When you don't use quotes: `enemy1` , you are referencing the `enemy1` **variable**.

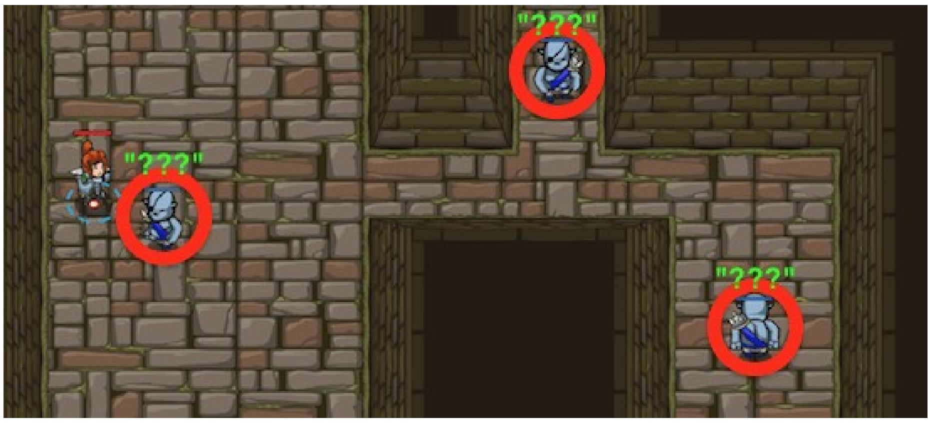
Known Enemy Solution

```
// You can use a variable like a nametag.

var enemy1 = "Kratt";
var enemy2 = "Gert";
var enemy3 = "Ursa";

hero.attack(enemy1);
hero.attack(enemy1);

hero.attack(enemy2);
hero.attack(enemy2);

hero.attack(enemy3);
hero.attack(enemy3);
```

#16. Master of Names

Level Overview and Solutions

Intro



Variables contain information to be referenced later. You can assign a new value to a variable any time you want.

Use `findNearestEnemy()` to target the nearest enemy.

```
var closestEnemy = hero.findNearestEnemy();
```

Default Code

```
// Your hero doesn't know these enemy's names!
// The glasses give you the findNearestEnemy ability.

var enemy1 = hero.findNearestEnemy();
hero.attack(enemy1);
hero.attack(enemy1);

var enemy2 = hero.findNearestEnemy();
hero.attack(enemy2);
hero.attack(enemy2);
```

Overview

Remember from the last level, **variables** are symbols that represent data. The variable's value can **vary** as you store new data in it, which is why it's called a variable.

Now instead of using the names of the enemies, you can use your glasses and the `findNearestEnemy()` method to store references to the ogres in variables. You don't need to use their names.

When you call the `findNearestEnemy()` method, you **must store the result in a variable**, like `enemy3` (you can name it whatever you want). The variable will remember what the nearest enemy **was** when you called the `findNearestEnemy()` method, so make sure to call it when you see a nearby enemy.

Remember: when you use quotes, like `"Kratt"` , you are making a **string**. When you don't use quotes, like `enemy1` , you are referencing the `enemy1` **variable**.

Ogre munchkins still take two hits to defeat.

Master of Names Solution

```
// Your hero doesn't know these enemy's names!
// The glasses give you the findNearestEnemy ability.

var enemy1 = hero.findNearestEnemy();
hero.attack(enemy1);
hero.attack(enemy1);

var enemy2 = hero.findNearestEnemy();
hero.attack(enemy2);
hero.attack(enemy2);

var enemy3 = hero.findNearestEnemy();
hero.attack(enemy3);
hero.attack(enemy3);
```

#16a. Lowly Kithmen

Level Overview and Solutions

Intro

Coming soon!

Default Code

```
// Create a second variable and attack it!

var enemy1 = hero.findNearestEnemy();
hero.attack(enemy1);
hero.attack(enemy1);
```

Overview

Because you don't know the names of these ogres, you can use the `findNearestEnemy` method from your glasses to store references to the ogres in variables.

When you call the `findNearestEnemy` method, you **must store the result in a variable**, like `enemy2` (you can name it whatever you want). The variable will remember what the nearest enemy **was** when you called the `findNearestEnemy` method, so make sure to call it when you see a nearby enemy.

Remember: when you use quotes, like `"Kratt"`, you are making a **string**. When you don't use quotes, like `enemy1`, you are referencing the `enemy1` **variable**.

Lowly Kithmen Solution

```
// Create a second variable and attack it!

var enemy1 = hero.findNearestEnemy();
hero.attack(enemy1);
hero.attack(enemy1);

enemy1 = hero.findNearestEnemy();
hero.attack(enemy1);
hero.attack(enemy1);

hero.moveDown();
hero.moveRight();
hero.moveRight();
```

#16b. Closing the Distance

Level Overview and Solutions

Intro

Coming soon!

Default Code

```
hero.moveRight();

// You should recognize this from the last level.
var enemy1 = hero.findNearestEnemy();
// Now attack enemy1.
```

Overview

Like in the last level, you can use the `findNearestEnemy` method from your glasses to store references to the ogres in variables.

When you call the `findNearestEnemy` method, you **must store the result in a variable**, like `enemy2` (you can name it whatever you want). The variable will remember what the nearest enemy **was** when you called the `findNearestEnemy` method, so make sure to call it when you see a nearby enemy.

Remember: when you use `findNearestEnemy`, you need to be in line of sight of an enemy, or you won't be able to see her.

This level introduces a new type of ogre, the ogre thrower, who does much more damage than the ogre munchkins you have been fighting. There are many more kinds of ogres awaiting you in future worlds.

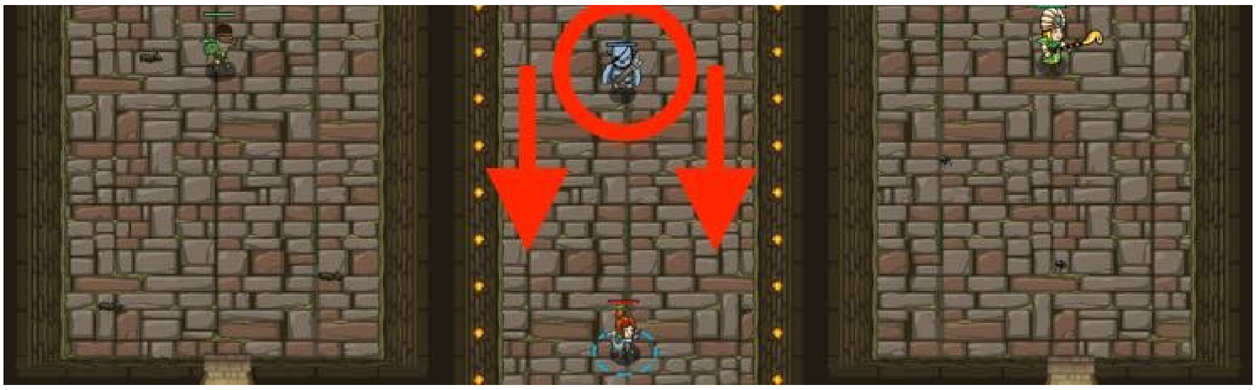
Closing the Distance Solution

```
hero.moveRight();

// You should recognize this from the last level.
var enemy1 = hero.findNearestEnemy();
// Now attack enemy1.

hero.attack(enemy1);
hero.attack(enemy1);

hero.moveRight();

enemy1 = hero.findNearestEnemy();
hero.attack(enemy1);

hero.moveRight();
```

#17. A Mayhem of Munchkins

Level Overview and Solutions

Intro



Use a **while-true loop** to fend off the attacking horde of munchkins.

Remember to use `findNearestEnemy()` :

```
while(true) {
    var enemy = hero.findNearestEnemy();
}
```

Default Code

```
// Inside a while-true loop, use findNearestEnemy() and attack!
```

Overview

In this level, you use a **while-true loop** to do two things:

First, use `findNearestEnemy()` to find an ogre. Remember to store the result in an `enemy` variable. Hover over the `findNearestEnemy()` method to see an example.

Then, `attack` using the `enemy` variable.

A Mayhem of Munchkins Solution

```
// Inside a while-true loop, use findNearestEnemy() and attack!

while (true) {
    var enemy = hero.findNearestEnemy();
    hero.attack(enemy);
    hero.attack(enemy);
}
```

#17a. The Skeleton

Level Overview and Solutions

Intro

Coming soon!

Default Code

```
// Use a loop to attack the skeleton.
// Its blunt sword hardly does any damage, but it has huge knockback.
```

Overview

Defeat the skeleton using a loop!

You can use the `findNearestEnemy` method to store the skeleton in a variable, then `attack` that variable. Put that in a loop, and your hero won't stop until the skeleton is defeated!

The skeleton has a very blunt sword, so it hardly does any damage. But given is gigantic size and strength, the sword alone is heavier than you are, so each time the skeleton hits you, you'll be knocked far into the air.

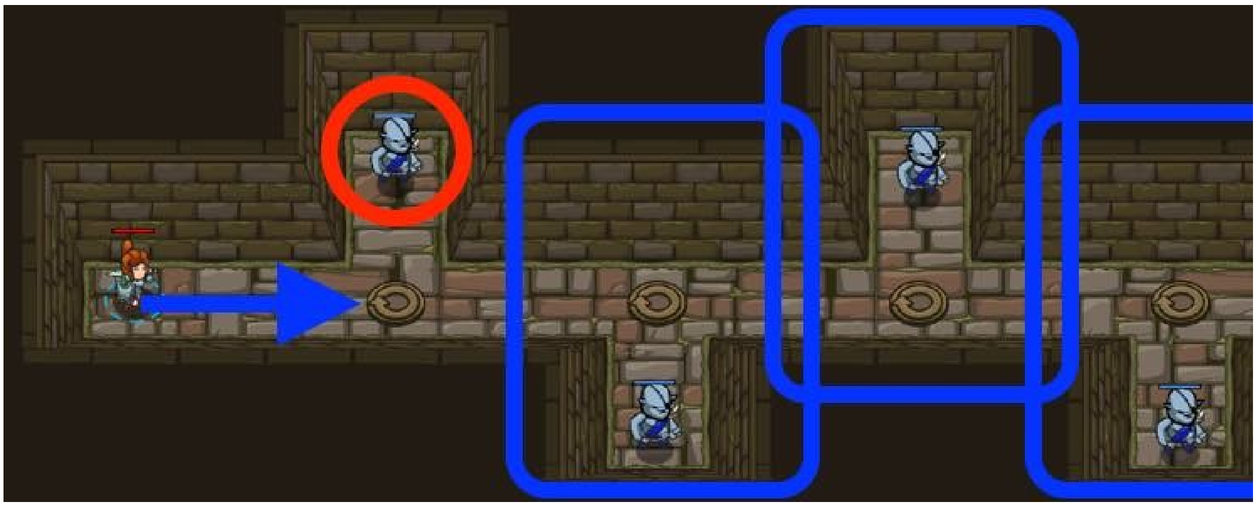
The Skeleton Solution

```
// Use a loop to attack the skeleton.
// Its blunt sword hardly does any damage, but it has huge knockback.

while(true) {
    hero.attack("Skully-Ton");
}
```

#18. The Gauntlet

Level Overview and Solutions

Intro



This level only needs **four** commands in a while-true loop.

Default Code

```
// Use what you've learned to defeat the ogres.
// Remember: it takes two attacks to defeat an ogre munchkin!
```

Overview

With your powers of looping and variables, it should be no sweat to take down all these munchkins. In fact, with the **while-true loop**, you can do it in just five lines of code:

1. one to start the while-true loop,
2. one to move to where you can see an enemy,
3. one to store the nearest enemy into a variable,
4. and two to attack,
5. because munchkins take two hits with your current sword

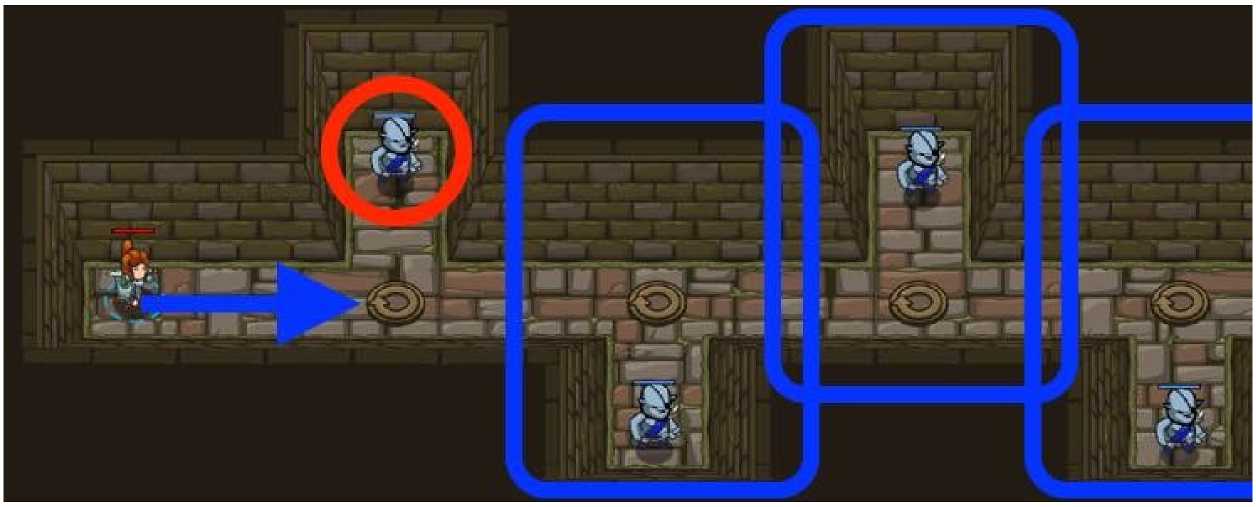
The Gauntlet Solution

```
// Use what you've learned to defeat the ogres.
// Remember: it takes two attacks to defeat an ogre munchkin!

while (true) {
    hero.moveRight();
    var enemy = hero.findNearestEnemy();
    hero.attack(enemy);
    hero.attack(enemy);
}
```

#18a. The Gauntlet A

Level Overview and Solutions

Intro



This level only needs **four** commands in a while-true loop.

Default Code

```
// Use what you've learned to defeat the ogres.
// Remember: it takes two attacks to defeat an ogre munchkin!
```

Overview

With your powers of looping and variables, it should be no sweat to take down all these munchkins. In fact, with the **while-true loop**, you can do it in just five lines of code:

1. one to start the while-true loop,
2. one to move to where you can see an enemy,
3. one to store the nearest enemy into a variable,
4. and two to attack,
5. because munchkins take two hits with your current sword

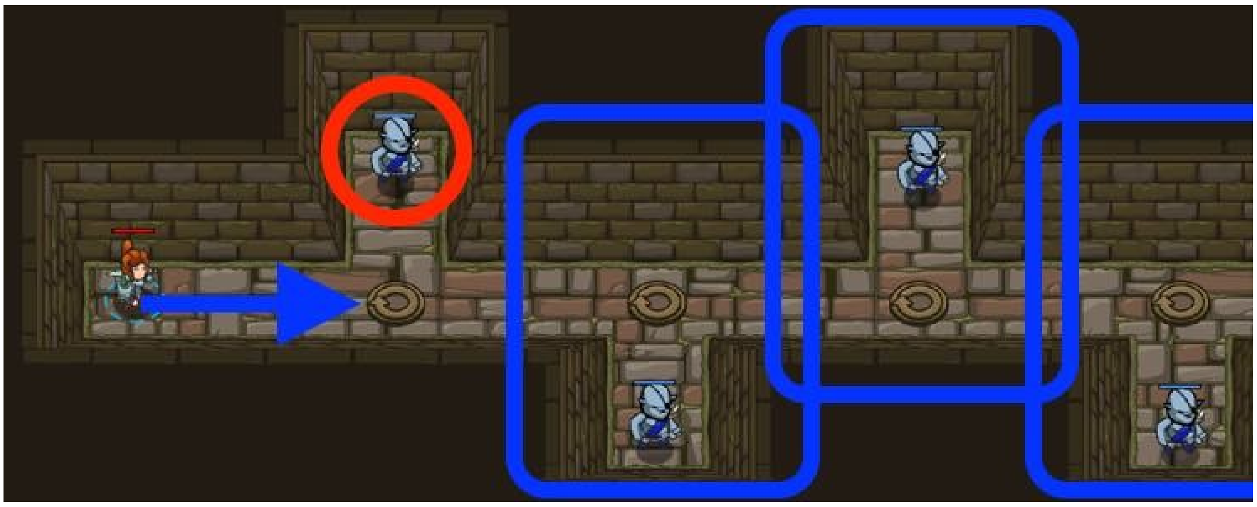
The Gauntlet A Solution

```
// Use what you've learned to defeat the ogres.
// Remember: it takes two attacks to defeat an ogre munchkin!

while (true) {
    hero.moveLeft();
    var enemy = hero.findNearestEnemy();
    hero.attack(enemy);
    hero.attack(enemy);
}
```

#18b. The Gauntlet B

Level Overview and Solutions

Intro



This level only needs **four** commands in a while-true loop.

Default Code

```
// Use what you've learned to defeat the ogres.
// Remember: it takes two attacks to defeat an ogre munchkin!
```

Overview

With your powers of looping and variables, it should be no sweat to take down all these munchkins. In fact, with the **while-true loop**, you can do it in just five lines of code:

1. one to start the while-true loop,
2. one to move to where you can see an enemy,
3. one to store the nearest enemy into a variable,
4. and two to attack,
5. because munchkins take two hits with your current sword

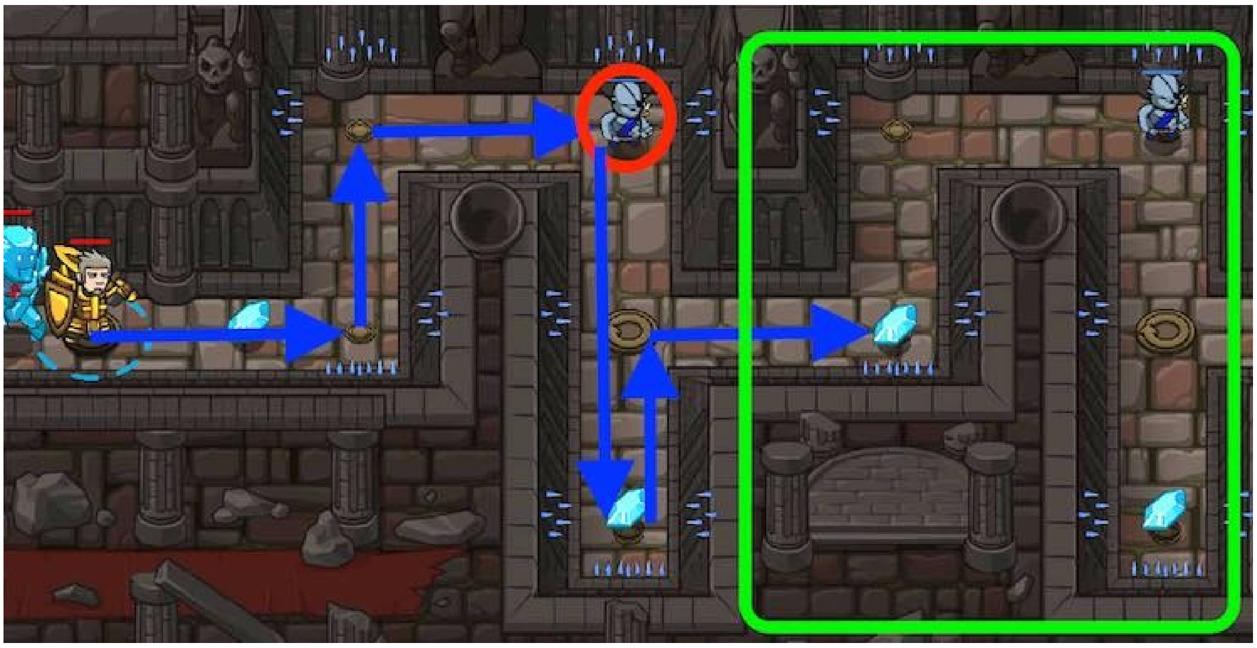
The Gauntlet B Solution

```
// Use what you've learned to defeat the ogres.
// Remember: it takes two attacks to defeat an ogre munchkin!

while (true) {
    hero.moveUp();
    var enemy = hero.findNearestEnemy();
    hero.attack(enemy);
    hero.attack(enemy);
}
```

#19. The Final Kithmaze

Level Overview and Solutions

Intro



Be sure to call `findNearestEnemy()` only when you can see an enemy.

Default Code

```
// Use a while-true loop to both move and attack.

while(true) {

}
```

Overview

This level combines **while-true loops** and **variables** to both solve a maze and attack enemies.

Now you see why you need variables, because you're actually going to vary the data in the variable. Inside your while-true loop, if you define an `enemy` variable, it will refer to each of the three ogre munchkins in the level as the loop repeats. Cool, huh?

Pay attention to where your while-true loop should repeat so that you don't move further than you need to.

Make sure that you call `findNearestEnemy()` when you can actually see the ogre munchkin with clear line of sight.

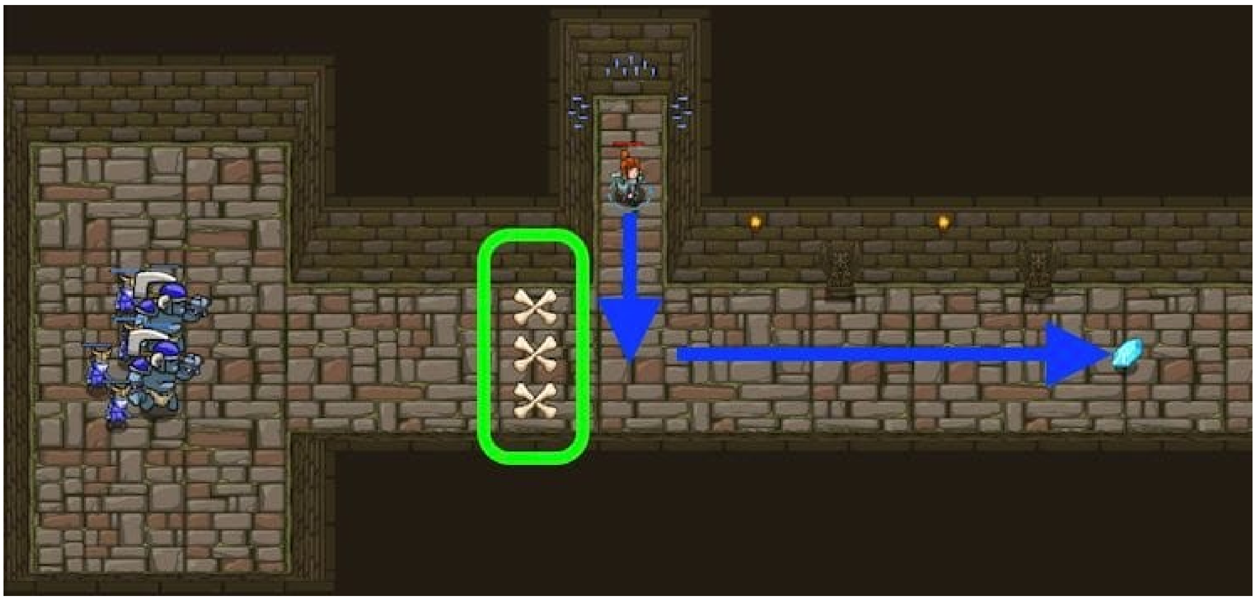
The Final Kithmaze Solution

```
// Use a while-true loop to both move and attack.

while (true) {
    hero.moveRight();
    hero.moveUp();
    var enemy = hero.findNearestEnemy();
    hero.attack(enemy);
    hero.attack(enemy);
    hero.moveRight();
    hero.moveDown(2);
    hero.moveUp();
}
```

#20. Kithgard Gates

Level Overview and Solutions

Intro



`buildXY("fence", x, y)` allows you to build a fence at a certain spot, like this:

```
hero.buildXY("fence", 40, 20);
```

Mouse over the map to find where you want to place the fence, and replace those numbers with the X and Y arguments of `buildXY`.

Default Code

```
// Build 3 fences to keep the ogres at bay!

hero.moveDown();
hero.buildXY("fence", 36, 34);
```

Overview

When you use a builder's hammer, instead of the `attack` method, you get the `buildXY` method. `buildXY` takes three arguments, instead of one: `buildType`, `x`, and `y`. So you can decide what to build and where to build it.

- `buildType`: either the string `"fence"`, to build fences, or the string `"fire-trap"`, to build fire traps.
- `x`: the horizontal position at which to build. You can **hover over the map** to find coordinates.
- `y`: the vertical position at which to build. `x` and `y` are both in meters.

`buildXY("fence", x, y)` allows you to build a fence at a certain spot, like this:

```
hero.buildXY("fence", 40, 20);
```

This level is **much easier to beat with `"fence"`** than with `"fire-trap"`. It's almost impossible to use fire traps to defeat the ogres. If you want to try it, fine, but it took us fifteen minutes to figure it out, and we built the level.

You only need to build three fences to stop the ogres and escape the dungeon to the right.

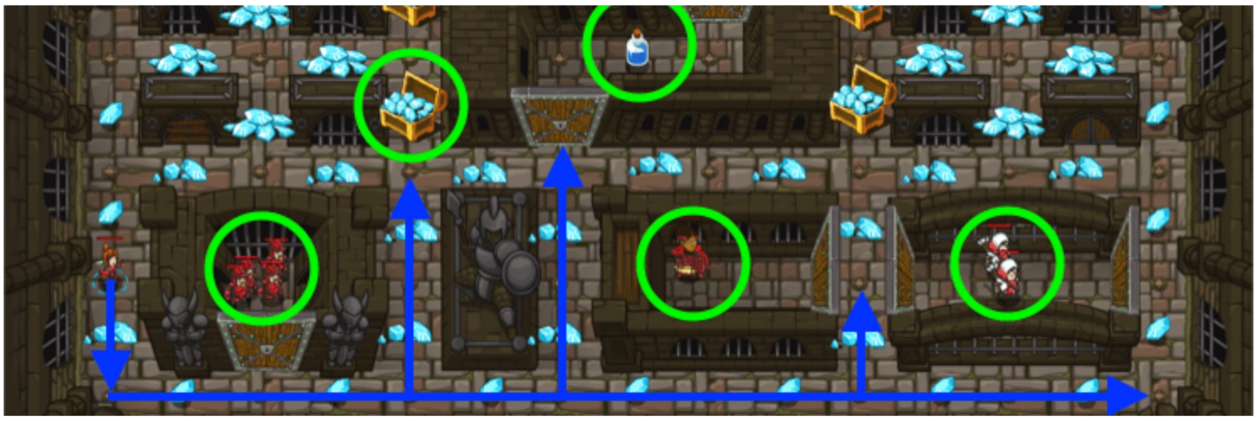
Kithgard Gates Solution

```
// Build 3 fences to keep the ogres at bay!

hero.moveDown();
hero.buildXY("fence", 36, 34);
hero.buildXY("fence", 36, 30);
hero.buildXY("fence", 36, 26);
hero.moveRight(3);
```

#21. Wakka Maul

Level Overview and Solutions

Intro



Walk over gems to collect them.

`say()` unit names to summon them.

`attack()` doors to break out your friends.

Default Code

```
// Welcome to Wakka Maul! Prepare for combat!
// Venture through the maze and pick up gems to fund your warchest.
// Break down doors to unleash allies (or enemies).
// For example, to attack the door labeled "g" use:
//hero.attack("g")
// If you have enough gold, you can call out for help by saying the type of unit you would like to
    summon!
//hero.say("soldier") to summon a Soldier at the cost of 20 gold!
//hero.say("archer") to summon an Archer at the cost of 25 gold!

hero.moveDown();
hero.moveRight();
hero.attack("g");
hero.say("soldier");
```

Overview

Wakka Maul

Battle your friends, coworkers and classmates in this all out brawl through the Kithgard dungeons!

Break out allies, summon more units, and evade the enemy's advances!

The doors are labelled `"a"`, `"b"`, `"c"`, `"d"`, `"e"`, `"f"`, `"g"`, `"h"`, `"i"`, `"j"`. Use these strings to attack the specific door you want!

The `human` side can summon `soldier` and `archer` while the `ogre` side can summon `scout` and `thrower`. All either side needs to do is to `say` the unit name, and have enough gems, to summon the units. To summon units you'll want to say their name:

```
// If on the human side:
hero.say("soldier"); // To summon a soldier for 20 gold.
hero.say("archer"); // To summon an archer for 25 gold.

// If on the ogre side:
hero.say("scout"); // To summon a scout for 18 gold.
hero.say("thrower"); // To summon 2 throwers for 9 gold each.
```

Wakka Maul Solution

```
// Welcome to Wakka Maul! Prepare for combat!
// Venture through the maze and pick up gems to fund your warchest.
// Break down doors to unleash allies (or enemies).
// For example, to attack the door labeled "g" use:
//hero.attack("g")
// If you have enough gold, you can call out for help by saying the type of unit you would like to
    summon!
//hero.say("soldier") to summon a Soldier at the cost of 20 gold!
//hero.say("archer") to summon an Archer at the cost of 25 gold!

hero.moveDown();
hero.moveRight();
hero.attack("g");
hero.moveRight(4);
hero.moveUp();
hero.attack("h");
hero.attack("i");
hero.moveUp(2);
while (true) {
    hero.say("archer");
}
```