DEPARTMENT OF
INFORMATION SYSTEMS
AND COMPUTER SCIENCE

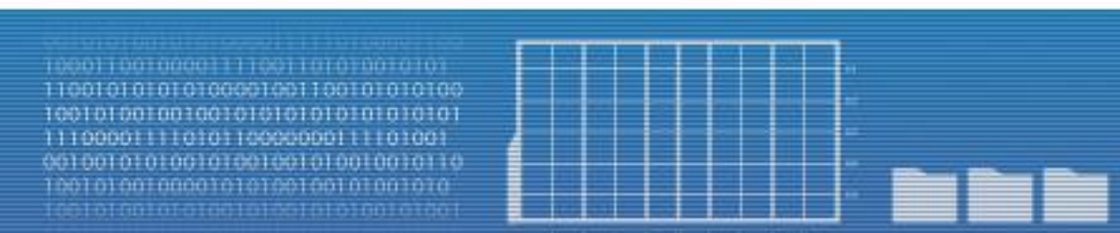
Learning C++

"Simple" Start

Welcome to Class!

- This is definitely a programming class (so expect to be coding A LOT)

- Our language of choice is C++
- And we use SFML for Graphics
- More on that in later lessons



The Catch

- The starting part might be a bit difficult for everyone as we find our footing
  - Starting with a new programming language is always difficult
  - But you'll get used to it after a while
  - …right?
  - I hope so!

The Catch

- This is not a C++ class
  - It is expected that the students have some coding experience already coming in

- Not everything is going to be explicitly stated in the slide sets
  - I'm sure you guys can do some of the simple stuff yourselves, right?

The Catch

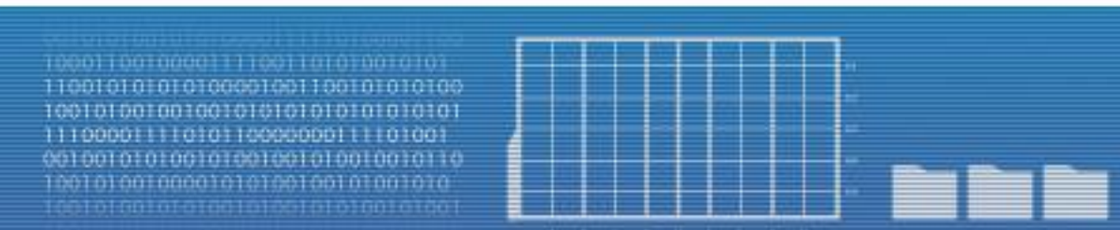
- Your best friends will be:
  - Documentation: http://www.cplusplus.com/doc/
  - Other sites: ~~StackOverflow~~
- Me, your teacher!
  - I'm **not** here to spoonfeed you code
  - But I **am** here to help you as needed

What You Need

- You'll need C++ to be able to compile your code on your local device
  - For the beginning portion, any version of C++ will do so you don't have to worry so much about it
- Feel free to use an online compiler for now, especially for the shorter exercises

What You Need

- We'll be doing set-up for SFML proper on our own machines in a later lesson
  - SFML is an extension to C++ so we'll be downloading it separately
- I strongly suggest going to their website and downloading your C++ version from there to prepare for it though



C++ and SFML

- I would like to request for us to all be using the 2.5.1 version of SFML

- Go to their website and download the matching C++ and SFML versions
  - https://www.sfml-dev.org/download/sfml/2.5.1/

C++ and SFML

Download SFML 2.5.1

On Windows, choosing 32 or 64-bit libraries should be based on which platform you want to compile for, not which OS you have. Indeed, you can perfectly compile and run a 32-bit program on a 64-bit Windows. So you'll most likely want to target 32-bit platforms, to have the largest possible audience. Choose 64-bit packages only if you have good reasons.

The compiler versions have to match 100%!

Here are links to the specific MinGW compiler versions used to build the provided packages:

TDM 5.1.0 (32-bit), MinGW Builds 7.3.0 (32-bit), MinGW Builds 7.3.0 (64-bit)

Visual C++ 15 (2017) - 32-bit	Download | 16.3 MB	Visual C++ 15 (2017) - 64-bit	Download | 18.0 MB
Visual C++ 14 (2015) - 32-bit	Download | 18.0 MB	Visual C++ 14 (2015) - 64-bit	Download | 19.9 MB
Visual C++ 12 (2013) - 32-bit	Download | 18.3 MB	Visual C++ 12 (2013) - 64-bit	Download | 20.3 MB
GCC 5.1.0 TDM (SJLJ) - Code::Blocks - 32-bit	Download | 14.1 MB		
GCC 7.3.0 MinGW (DW2) - 32-bit	Download | 15.5 MB	GCC 7.3.0 MinGW (SEH) - 64-bit	Download | 16.5 MB

C++ and SFML

Download SFML 2.5.1

On Windows, choosing 32 or 64-bit libraries should be based on which platform you want to compile for, not which OS you have. Indeed, you can perfectly compile and run a 32-bit program on a 64-bit Windows. So you'll most likely want to target 32-bit platforms, to have the largest possible audience. Choose 64-bit packages only if you have good reasons.

The compiler versions have to match 100%!

Here are links to the specific MinGW compiler versions used to build the provided packages:

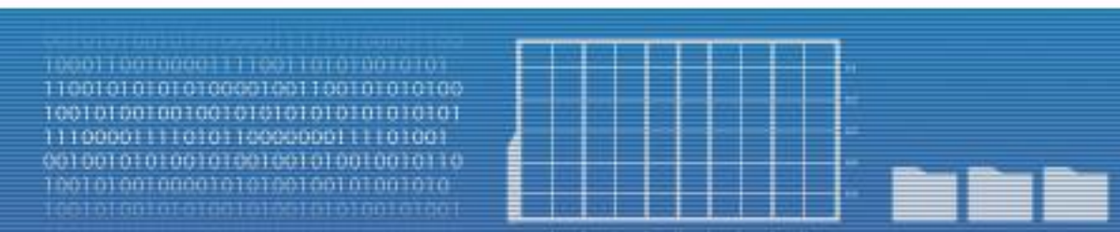
TDM 5.1.0 (32-bit), MinGW Builds 7.3.0 (32-bit), MinGW Builds 7.3.0 (64-bit)

Please pay special attention to this!

Visual C++ 15 (2017) - 32-bit	Download | 16.3 MB	Visual C++ 15 (2017) - 64-bit	Download | 18.0 MB
Visual C++ 14 (2015) - 32-bit	Download | 18.0 MB	Visual C++ 14 (2015) - 64-bit	Download | 19.9 MB
Visual C++ 12 (2013) - 32-bit	Download | 18.3 MB	Visual C++ 12 (2013) - 64-bit	Download | 20.3 MB
GCC 5.1.0 TDM (SJLJ) - Code::Blocks - 32-bit	Download | 14.1 MB		
GCC 7.3.0 MinGW (DW2) - 32-bit	Download | 15.5 MB	GCC 7.3.0 MinGW (SEH) - 64-bit	Download | 16.5 MB

C++ and SFML

- Compiler versions need to match
- This means that the C++ you download should be from the link they specify
- And your SFML version should be the matching one for that



C++ and SFML

- Things to take note of:
  - If your C++ compiler is 32-bit, your SFML should also be the 32-bit version
  - Similarly, if your C++ compiler is 64-bit, your SFML should also be the 64-bit version

C++ and SFML

- Things to take note of:
  - There is a separate version for Linux users
  - For Mac users, I strongly suggest using the app “Xcode” since it has support for SFML
    - To those who need help with this specifically, please send me an email and I will try to guide you appropriately

Exercises

- These are simple tasks that I suggest you do to get started and practiced

- As a general rule, Exercises are NOT going to be checked or graded
  - But please believe me that they are picked out and included as stepping stones so there is still merit in doing them

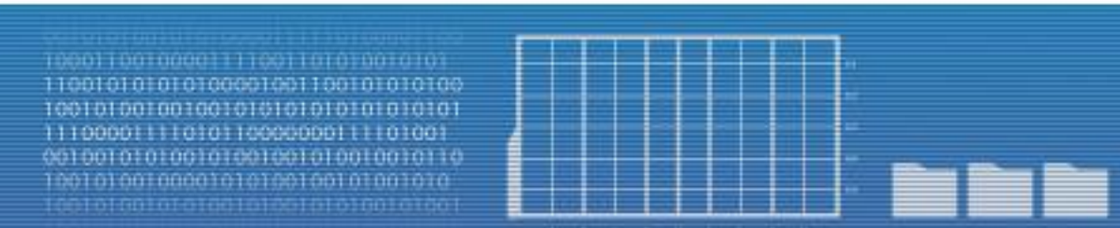
Exercise #1

- Create a C++ program that accepts user input and echoes it back
  - User input:

```
Hello there!
```

- Program output:

```
Hello there!
```



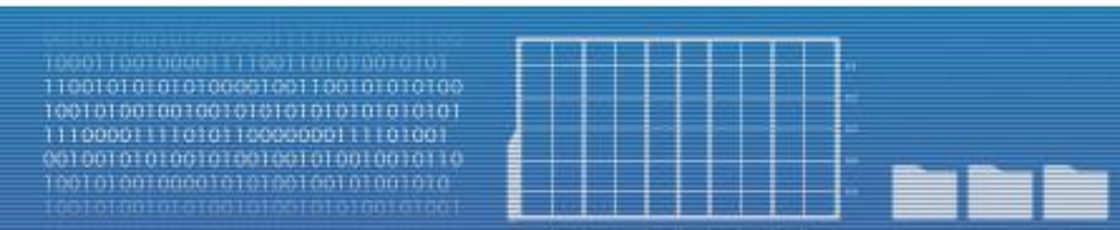
Exercise #2

- Create a C++ program that accepts user input and adds the two numbers given
  - User input:

```
12 24
```

- Program output:

```
36
```



DISCS

Exercise #3

- Create a C++ program that reads numbers from a file and returns their sum
  - File contents:

```
1 2 3 4 5 6 7 8 9
```

- Program output:

```
45
```

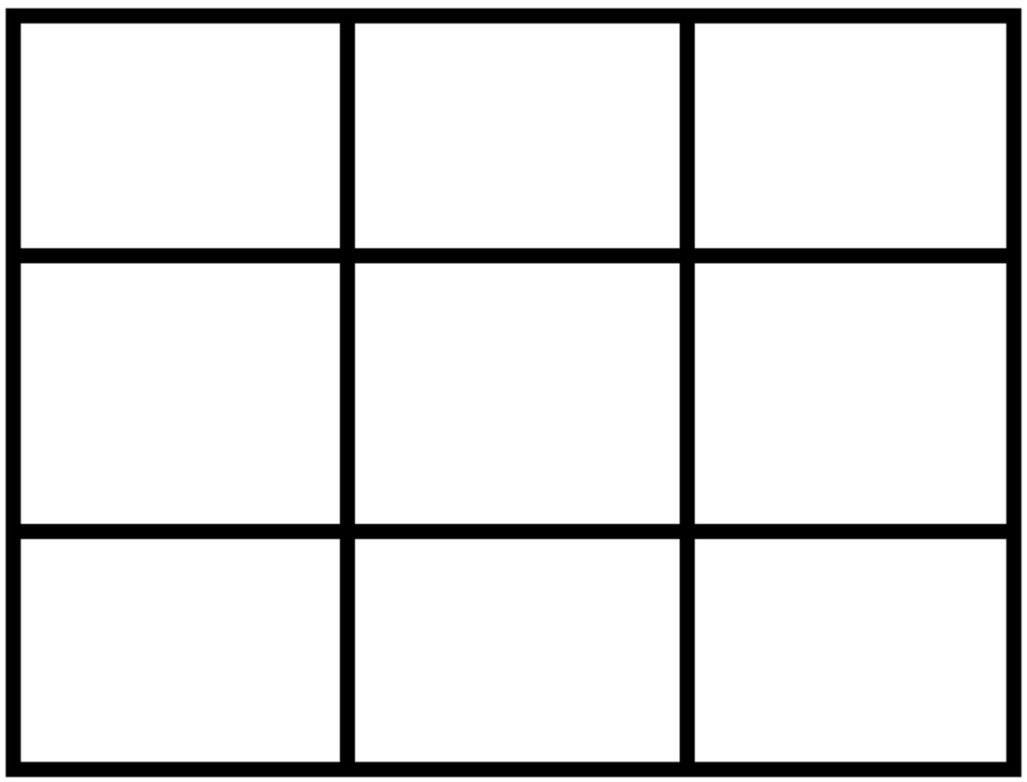
Groupings

- Homework is to be in groups of **three**
  - **This grouping is final for the semester**

- The Final Project will be done in groups, but I normally allow for students to work solo if they request it **and** have proved capable enough

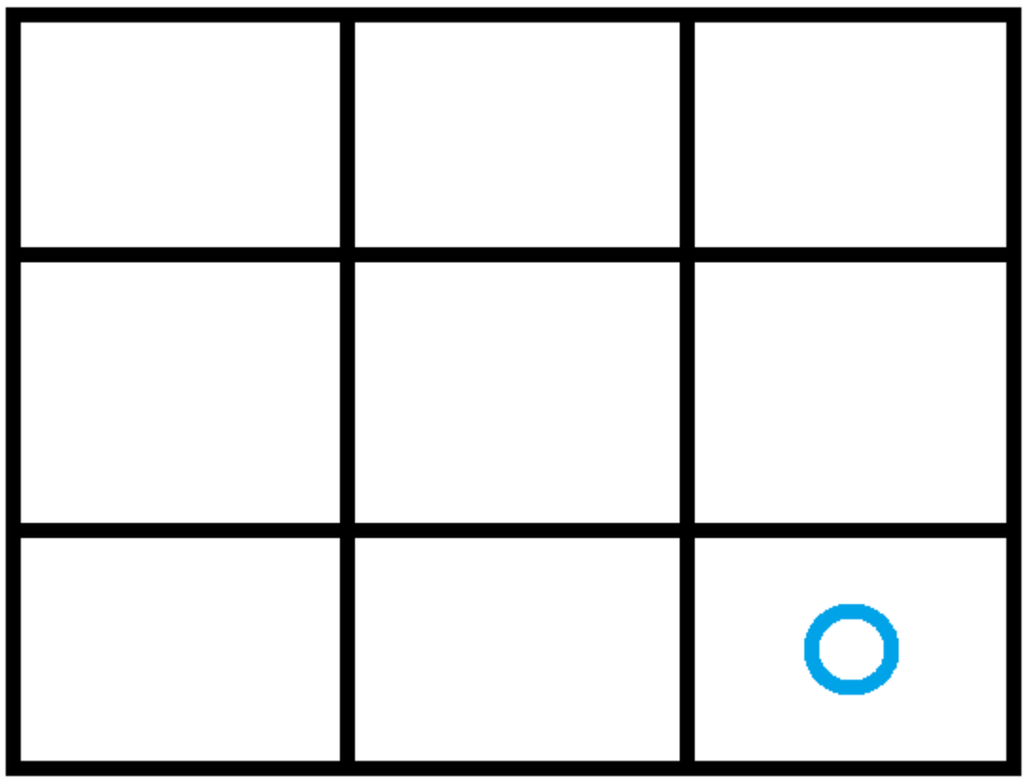
Homework

- Imagine a world represented by a 3x3 grid:



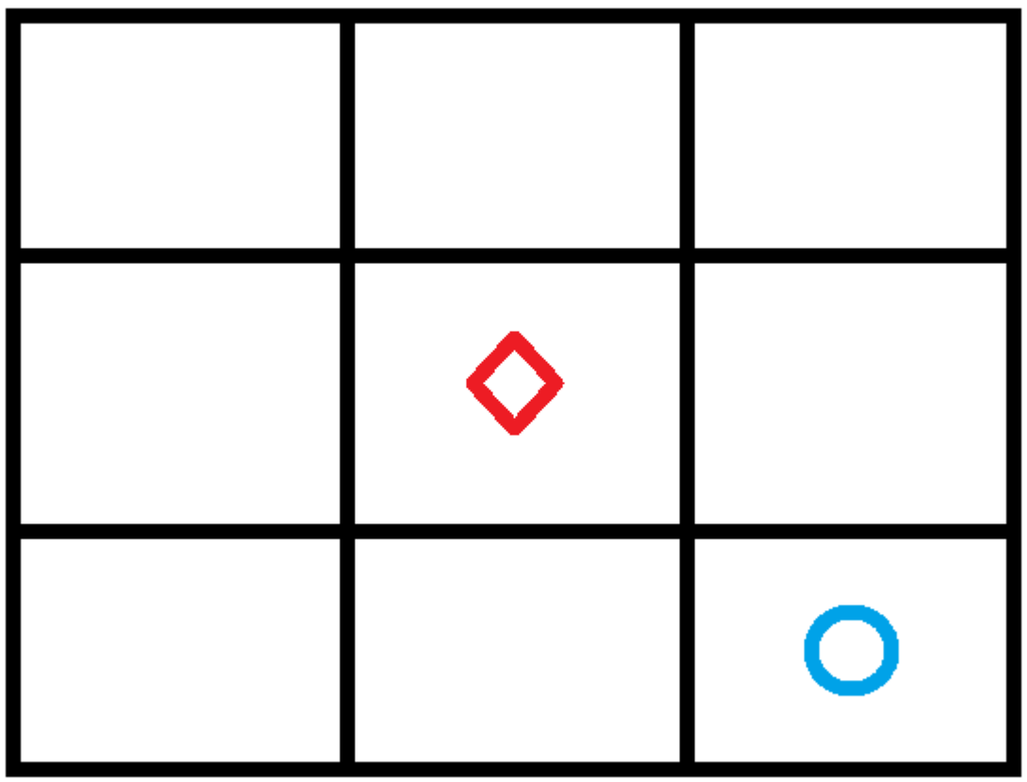
Homework

- You start here...



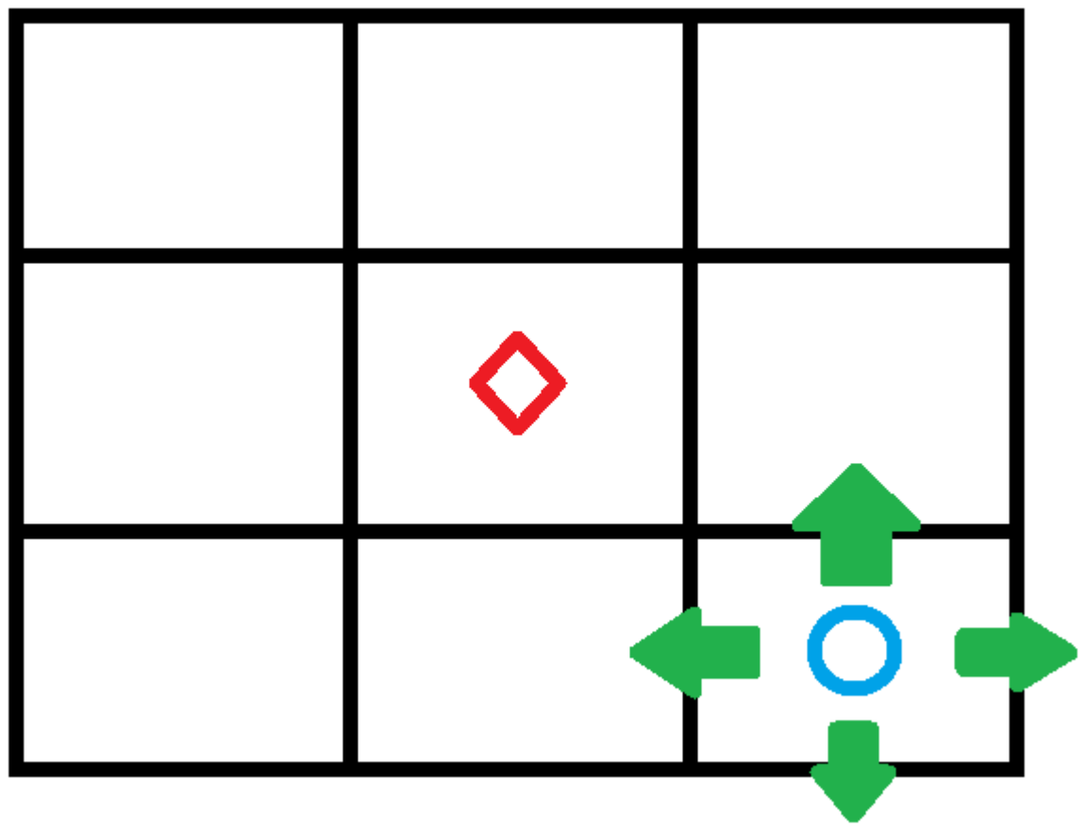
Homework

- ... and must eliminate a target found here:



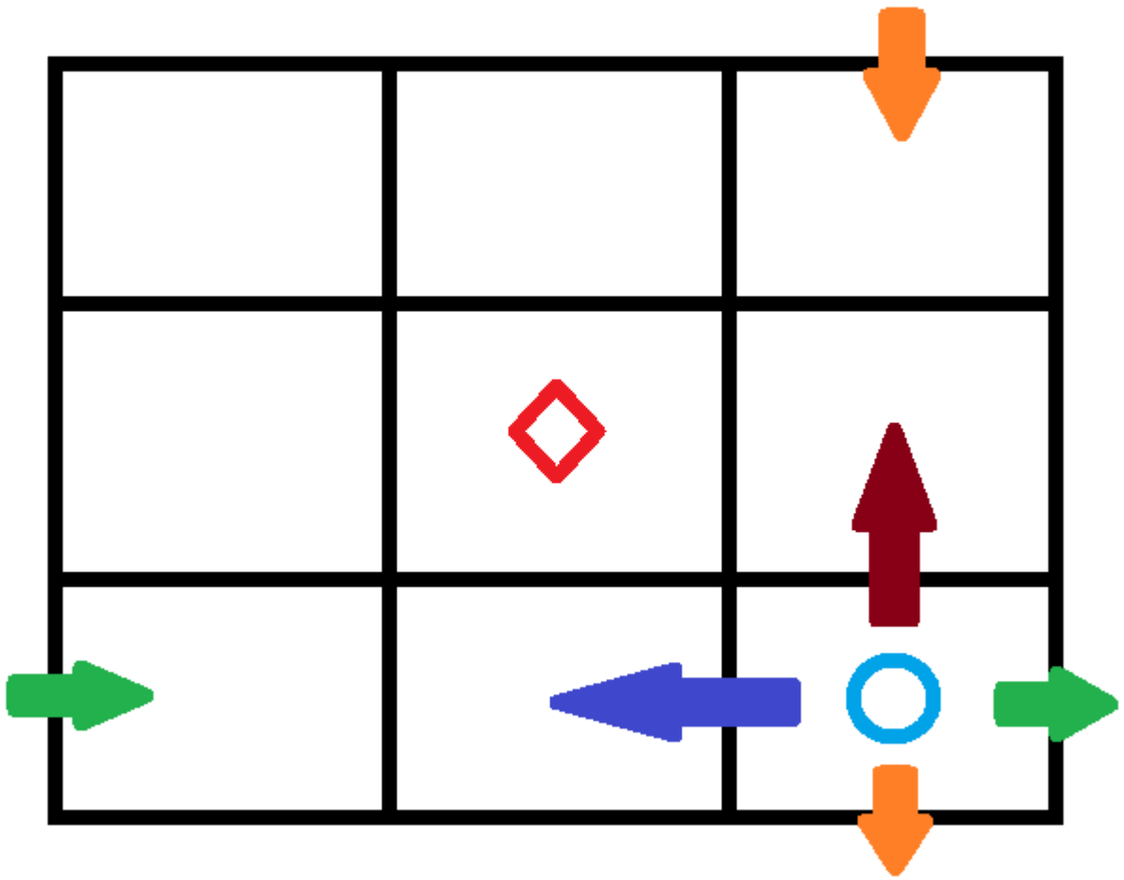
Homework

- You can move in any of the four cardinal directions: `north`, `east`, `west`, and `south`



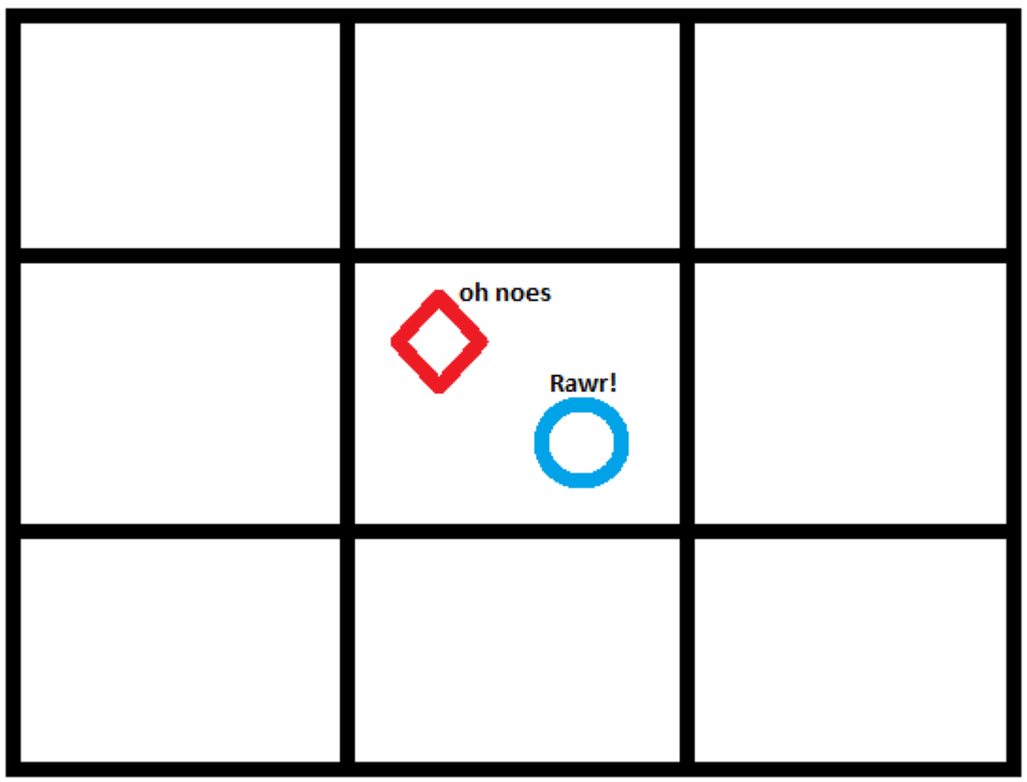
Homework

- But note that the world has no walls and actually wraps around like a Karnaugh map:



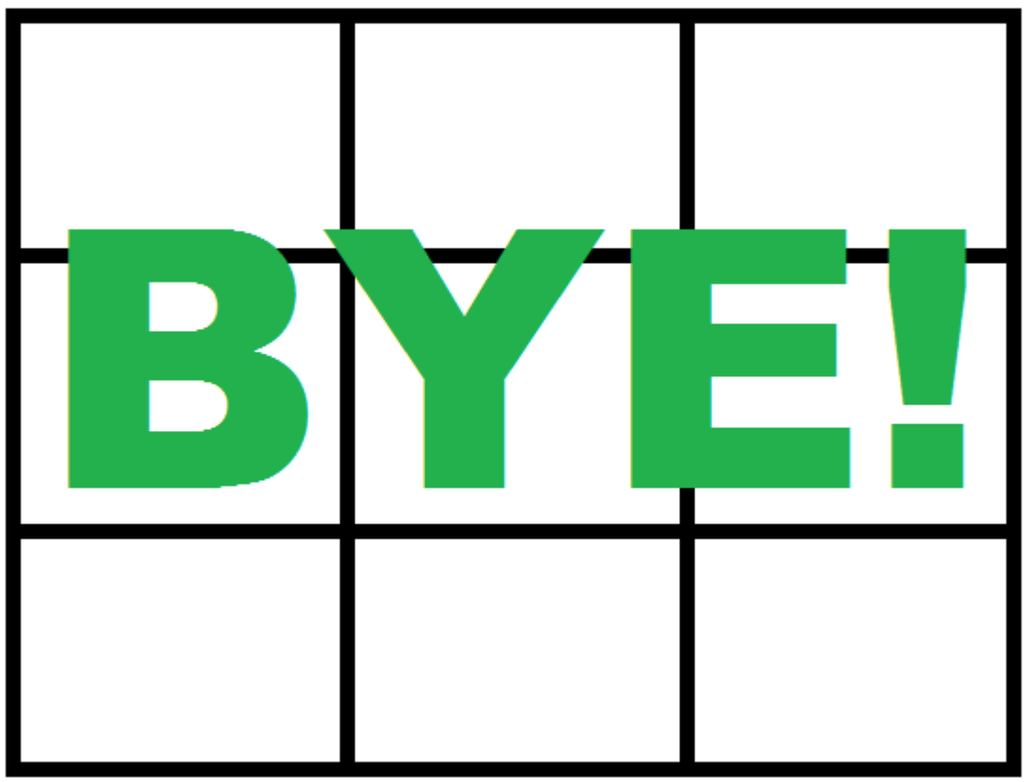
Homework

- You can also `attack` if you're in the same room as your target:



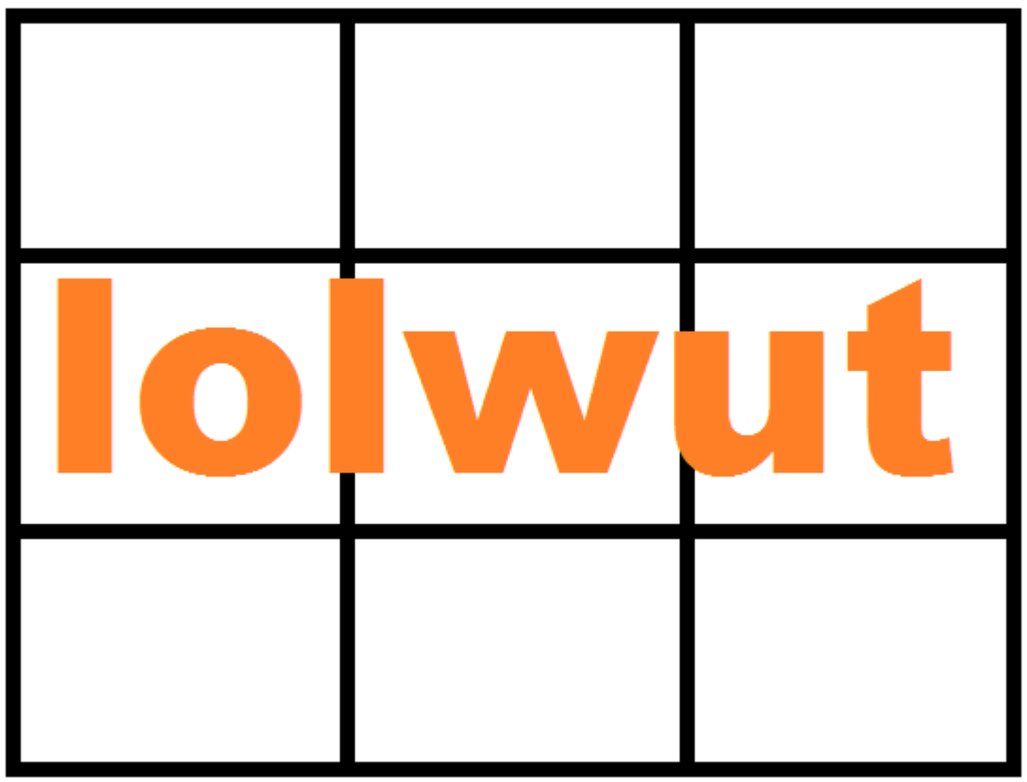
Homework

- And of course, you can `exit` the game
  - Attacking the target should trigger an exit also



Homework

- Any other input is to be treated as an invalid command

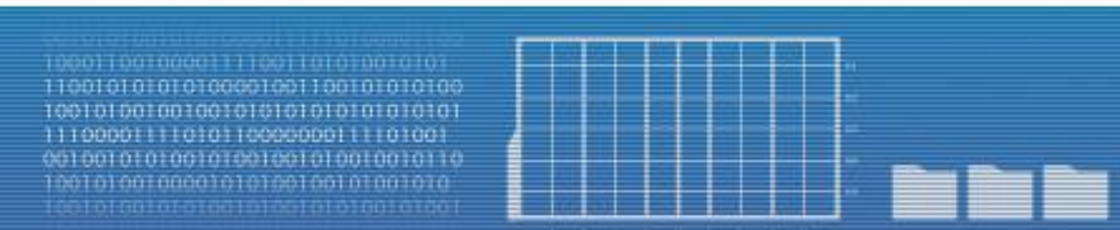


Homework

- Your homework is to make a text-based game based on the specs above
- It is up to you how you wish to display the world to the player
- You can use a simple MUD setup wherein you can only see whatever is in the same room as you
  - https://en.wikipedia.org/wiki/MUD

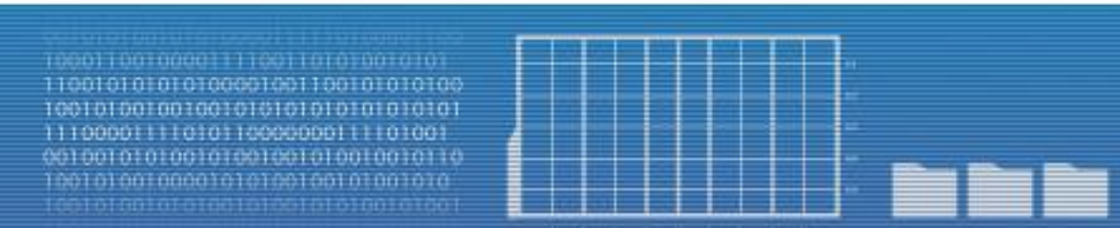
Homework

- Regardless, you must have an option to display the world and the positions of your character and your target at all times
- Can be always on or simply toggled either at the start via command-line argument or at anytime during the game via a command
  - Example: Omniscient setup



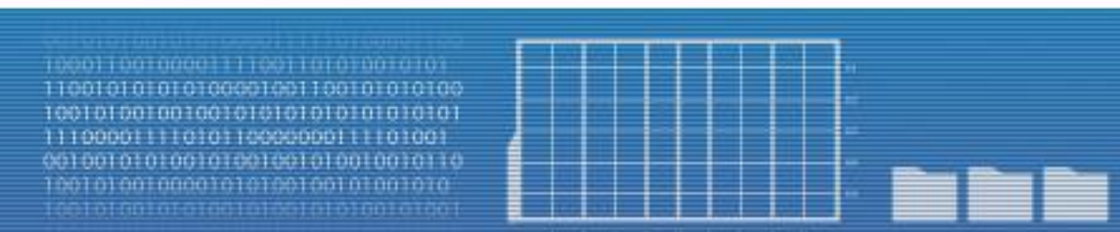
Homework

- Note that commands are not allowed to be chained (example: `north north` or `n n` should not trigger going north twice and should be treated as an invalid command)



Homework

- Your program must be scalable
- The following should be changeable by simply editing the contents of the corresponding variable:
  - Size of the world (W x H grid)
  - Player starting position
  - Target starting position



Homework

- These variables will be read from a file named “settings.txt” and loaded into your game upon start-up

- The rest of the game runs out of the console, simulating a real player
  - See the sample executables for an example of how it should run