CMPM 120

Time, Procedural Generation

Objectives

By the end of today you should be able to...

1. Time
    a. Describe how Phaser handles **timers** and **events**
    b. Explain **callbacks** and **closures**
    c. Implement an **event that loops**
2. Your Endless Runner
    a. Have answers to your questions about your endless runner
3. Procedural Generation
    a. Describe several ways to implement **random content** in a game

Revising Past Assignments

Since the point of the exercises is to measure your understanding of the material, I will allow you to submit a revision of your past assignment as long as:

- Your updated submission demonstrates your understanding of the material
  - Include lots of comments, explaining why you chose to implement your solution in that way
  - If I can't understand why you made your decisions, you don't get the points
- Late penalties still apply, but from the point of your original turn-in
  - I want to encourage you to turn stuff in on time
  - Turning stuff in late makes extra work for both of us
- Revision grading will happen at a time of my discretion
- No revisions will be accepted past the end of week 9
- **Does not apply to the final project:** the final project milestones are **hard deadlines**

If your files aren't updating...

Empty cache and hard refresh

Safari

Enable the Develop menu from Safari menu - Preferences - Advanced.

On Safari version 11.1 and above: CMD+OPTION+R reloads the page ignoring cache.

On Safari version 9 and above: CMD+SHIFT+R reloads the page ignoring cache.

https://superuser.com/questions/186594/how-can-i-force-safari-to-perform-a-full-page-reload-without-using-the-mouse

Chrome

Windows: Ctrl + the Reload button. Or Ctrl + F5.

Or open the Chrome Dev Tools by pressing F12. Right click on the refresh button, select from menu.

Mac: ⇧ Shift + the Reload button. Or ⌘ Cmd + ⇧ Shift key + R.

Firefox

Open the developer toolbox (Ctrl+Shift+I or Cmd+Opt+I on Mac). Click the settings button (near the top right). Scroll down to the Advanced settings on the bottom right. Check the option "Disable Cache (when toolbox is open)". https://support.mozilla.org/en-US/questions/1103414

Or Ctrl + F5, or Ctrl + Shift + R, or ⌘ Cmd + ⇧ Shift key + R.

Checking that your code is correct

Linter

What is a linter?

lint, or a linter, is a tool that analyzes source code to flag programming errors, bugs, stylistic errors, and suspicious constructs. The term originates from a Unix utility that examined C language source code.

https://en.wikipedia.org/wiki/Lint_(software)

Javascript Linters

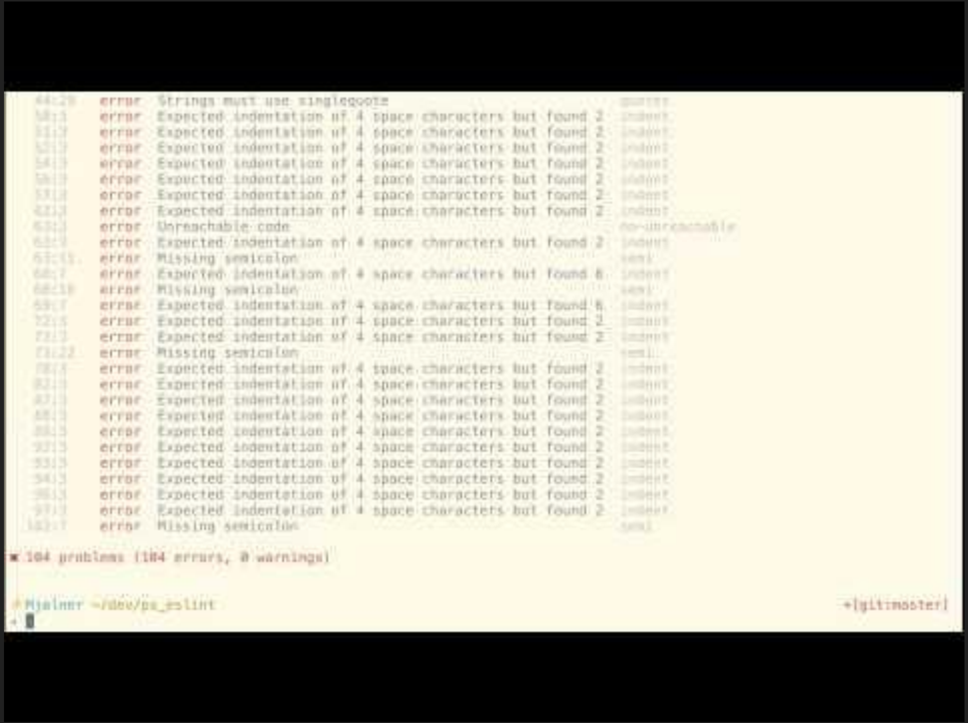
ESLint: https://eslint.org/

JSHint: https://jshint.com/

For Atom: https://atom.io/packages/linter-jslint

For Sublime:
https://packagecontrol.io/packages/SublimeLinter

Online: https://www.jslint.com/



```
  44:29      error  Strings must use singlequote                           quotes
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Unreachable code                                         no-unreachable
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Missing semicolon                                        semi
  [illegible]  error  Expected indentation of 4 space characters but found 6  indent
  [illegible]  error  Missing semicolon                                        semi
  [illegible]  error  Expected indentation of 4 space characters but found 6  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Missing semicolon                                        semi
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Expected indentation of 4 space characters but found 2  indent
  [illegible]  error  Missing semicolon                                        semi

✖ 104 problems (104 errors, 0 warnings)

Mjalner ~/dev/ps_eslint                                                  ±[git:master]
→
```

Events and Callbacks

Time

How do we make an event happen in the game?

- ???

Many different ways to solve the problem

```
Play.update() {
    doTheThing(); // use the game state update()
}

// create our own object and use its update()
ObstacleManager.update() { doTheThing(); }

var timer = game.timer.create(); // use a timer
timer.add(doTheThing);
timer.repeat(doTheThing);
timer.loop(doTheThing);
```

Phaser has a lot of tools to manage time

http://localhost:8000/time/timer_example.html

`game.time`	(the Time object)
`Timers`	(objects for individual timers)
`Timer Event`	(object that represents a single time-related event)

TimerEvent

```
new TimerEvent(timer, delay, tick, repeatCount, loop,
callback, callbackContext, arguments)
```

The timer object to use	timer
The delay before the event fires	delay
The next game clock time to fire at	tick
Repeat this many times	repeatCount
Does it loop?	loop
Function to call when it happens	callback
The value of `this` for the callback	callbackContext
Parameters for the callback function	arguments

...but that's complicated so let's simplify

```
timer.add(delay, callback, callbackContext, arguments);
```

The timer object to use	timer
The delay before the event fires	delay
Function to call when it happens	callback
The value of `this` for the callback	callbackContext
Parameters for the callback function	arguments

```
timer.loop(delay, callback, callbackContext, arguments);
timer.repeat(delay, repeatCount, callback, callbackContext,
arguments);
```

Callback Context

```
event.callback.apply(event.callbackContext, event.args);
```

https://github.com/photonstorm/phaser-ce/blob/da7bdf93b52ff1fb889612f03ef47293ec6af6ba/src/time/Timer.js#L478

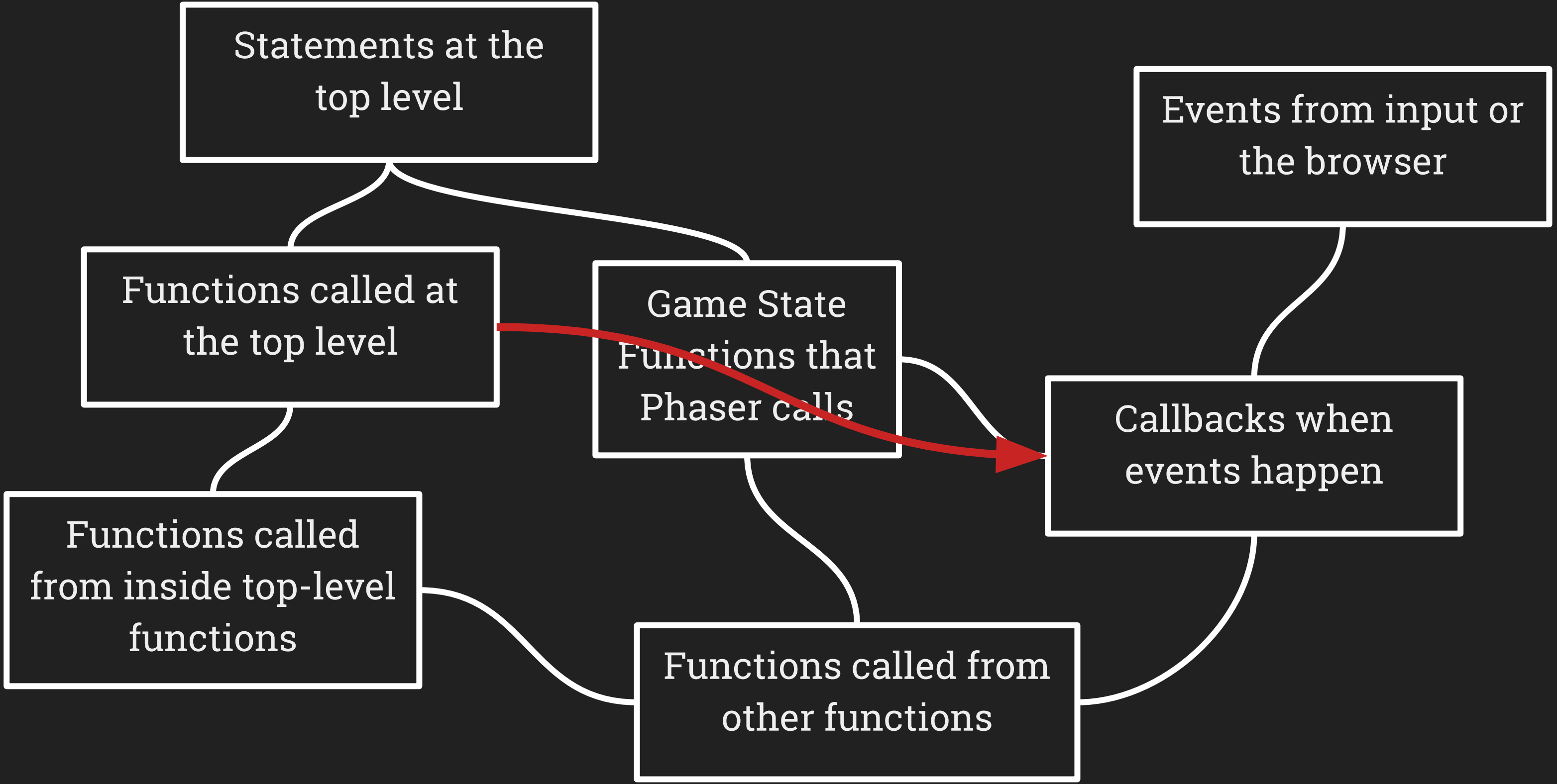
function.prototype.apply(thisArg, [argsArray])

The apply() method calls a function with a given `this` value, and arguments provided as an array (or an array-like object).

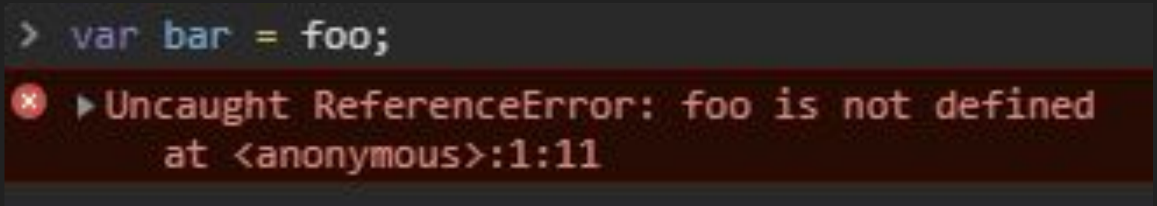
Many different ways to solve the problem

```
Play.update() {
    doTheThing(); // use the game state update()
}

// create our own object and use its update()
ObstacleManager.update() { doTheThing(); }

var timer = game.timer.create(); // use a timer
timer.add(doTheThing);
timer.repeat(doTheThing);
timer.loop(doTheThing);
```

When does the code run?


Statements at the top level
Functions called at the top level
Game State Functions that Phaser calls
Events from input or the browser
Callbacks when events happen
Functions called from inside top-level functions
Functions called from other functions

Function Scope



```
> var bar = foo;
✖ ▸Uncaught ReferenceError: foo is not defined
      at <anonymous>:1:11
```

Variable bindings are only valid in part of the program.

This region is called the **scope**.

let versus var

```
function exampleFunctionOne() {
    let first = 7;
    console.log(first);
    for(let first = 0; first < 5;
first++) {
        console.log(first);
    }
    console.log(first);
}
```

The let statement declares
an **enclosing block scope** local variable.

```
function exampleFunctionTwo() {
    // hoisting: var second;
    console.log(second);
    for(var second = 0; second < 5;
second++) {
        console.log(second);
    }
    console.log(second);
}
```

The var statement declares
a **function scope** variable.

Lexical Scope versus Closures

```
function parent() {
    var parent_value = 1;
    function child() {
        var child_value = 2;
        console.log(parent_value);
    }
    // error!
    console.log(child_value);
}
```

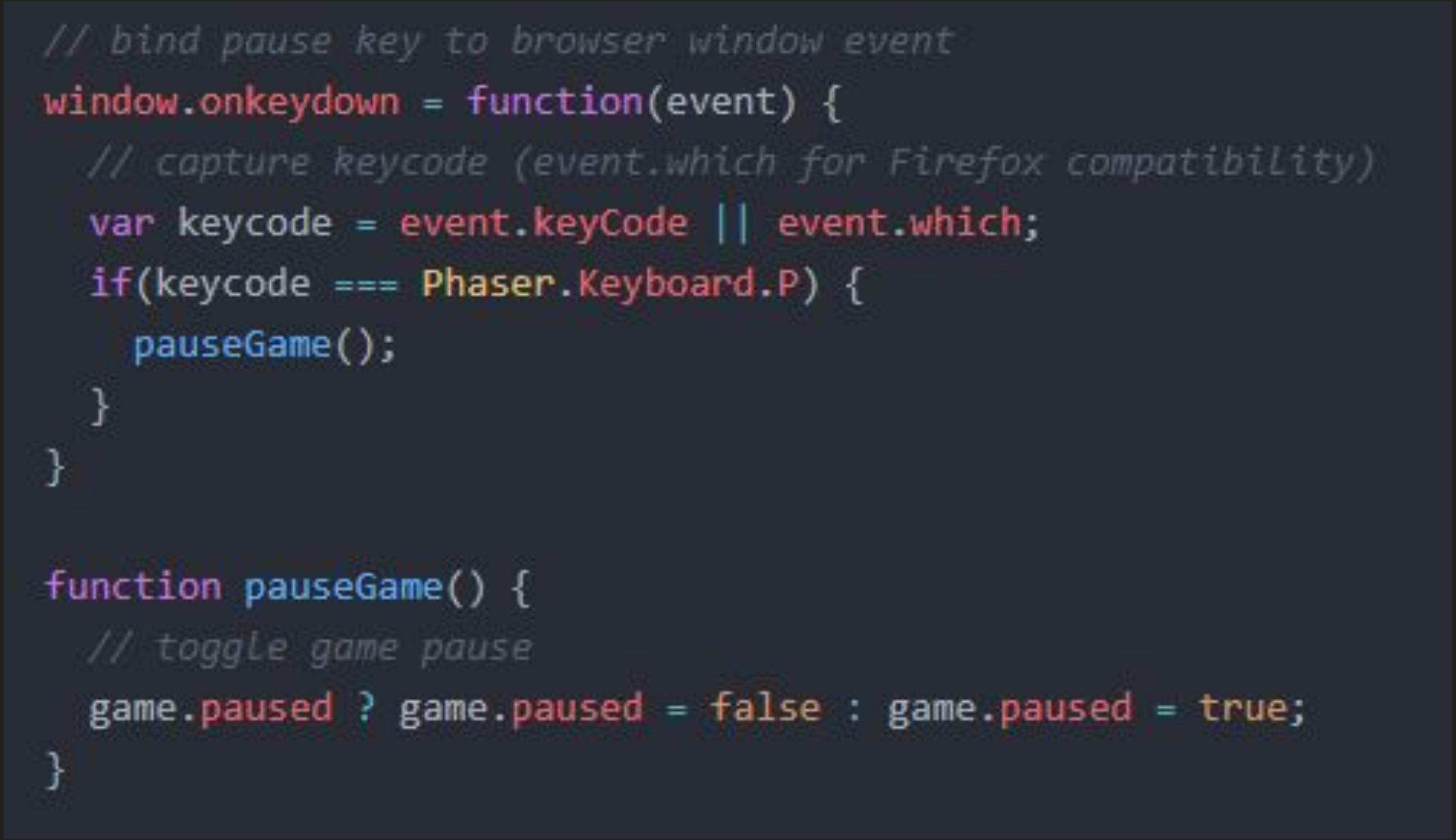
Lexical scope exists in the written code: the `parent_value` is accessible in the child function, but the `child_value` isn't accessible in the parent function.

```
function makeAdder(x) {
  return function(y) {
    return x + y;
  };
}
var add5 = makeAdder(5);
var add10 = makeAdder(10);
console.log(add5(2));  // 7
console.log(add10(2)); // 12
```

Closures use the run-time context from when the outer function was called and the inner function was created.

⚠️ BE CAREFUL WITH PAUSING ⚠️

Phaser does not call `update()` when paused. As a result, any input management tied to update will no longer function. For instance, if you bind pause to the P key, that key will turn pause on, but then be unable to turn pause off. REAL COOL GAME.



```
// bind pause key to browser window event
window.onkeydown = function(event) {
  // capture keycode (event.which for Firefox compatibility)
  var keycode = event.keyCode || event.which;
  if(keycode === Phaser.Keyboard.P) {
    pauseGame();
  }
}

function pauseGame() {
  // toggle game pause
  game.paused ? game.paused = false : game.paused = true;
}
```

Your endless runners

Some brief notes on

Procedural Generation

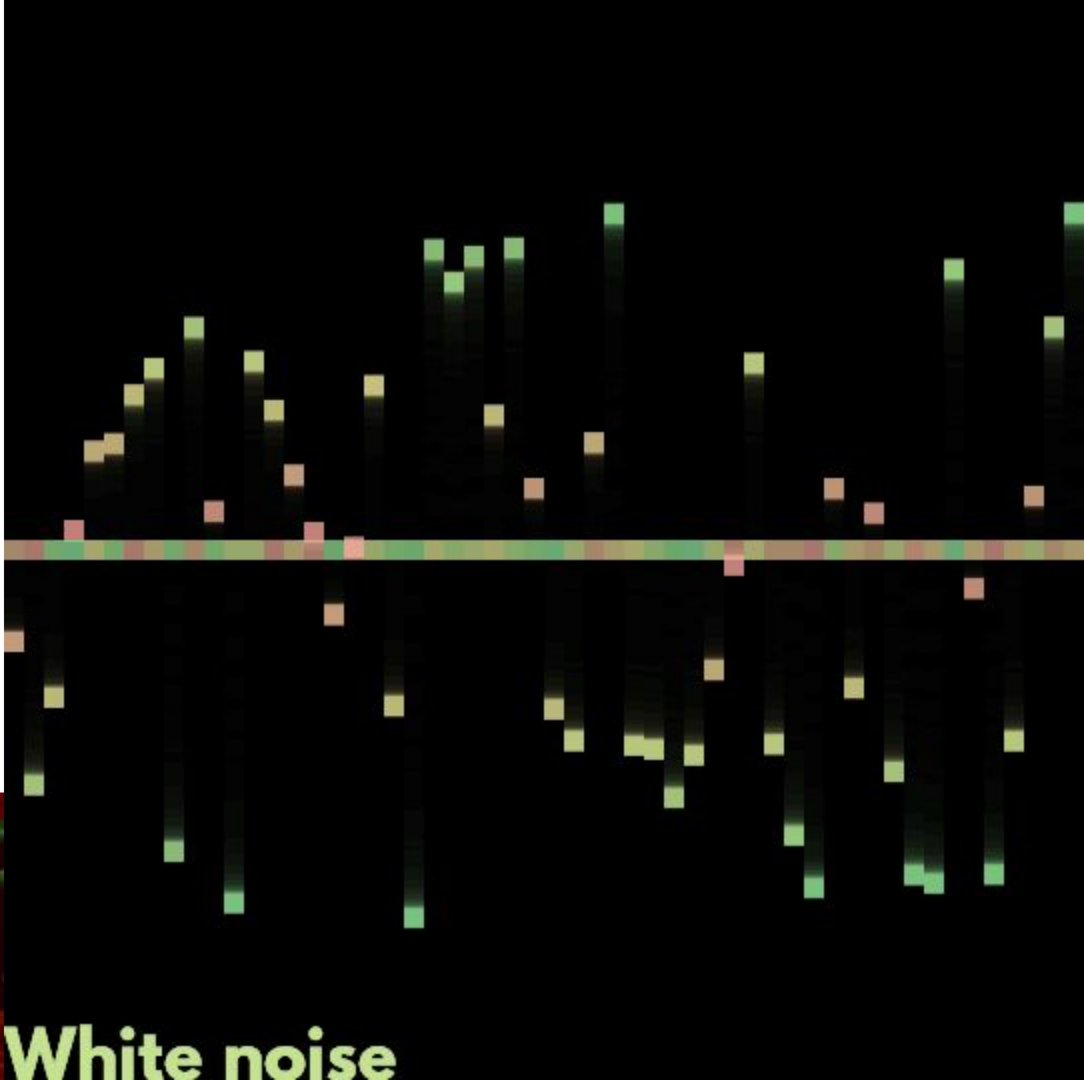
II. Noise

Noise

The most basic generative technique: use a random number.

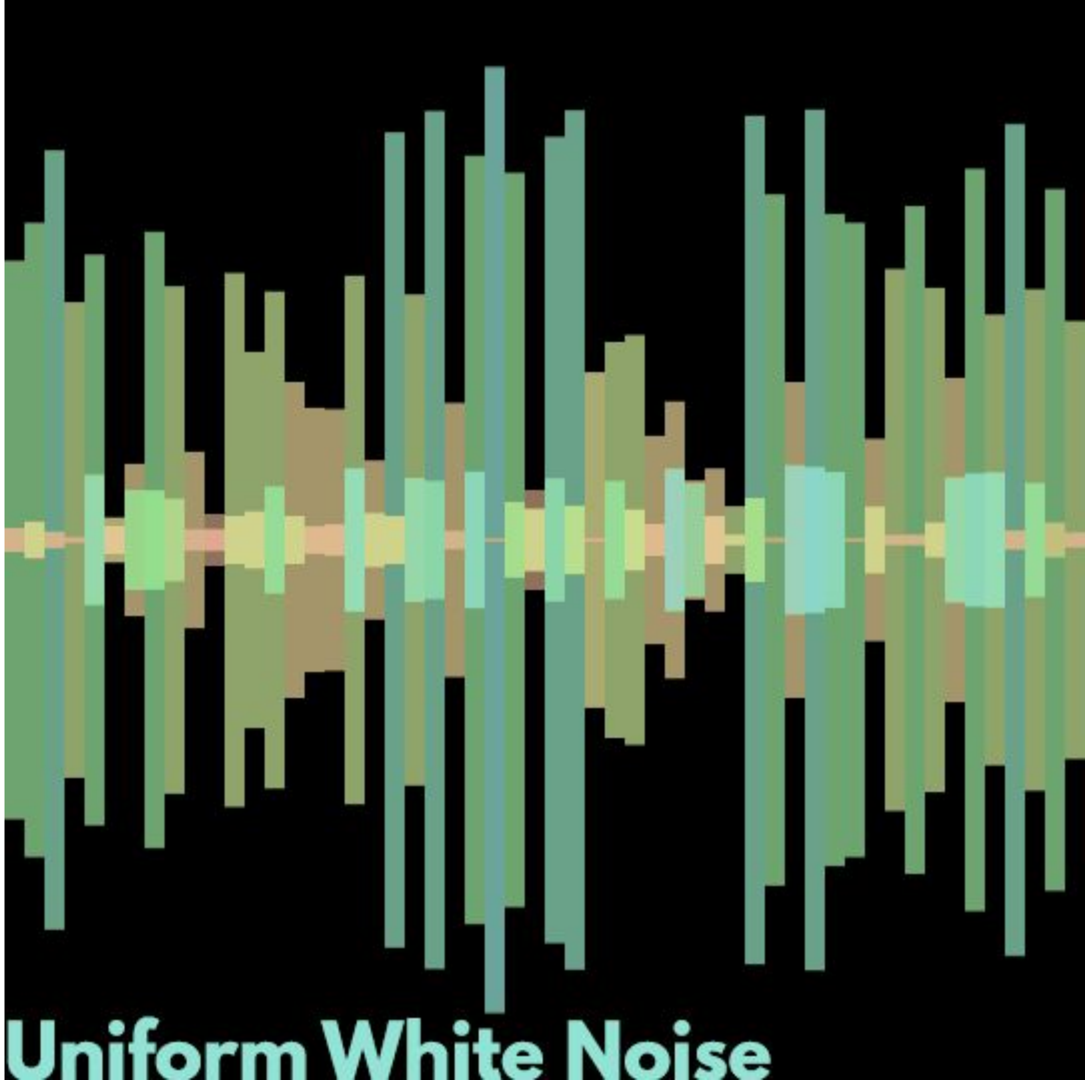
This is basically the same as rolling a single die.

(More on noise: https://www.redblobgames.com/articles/noise/introduction.html)



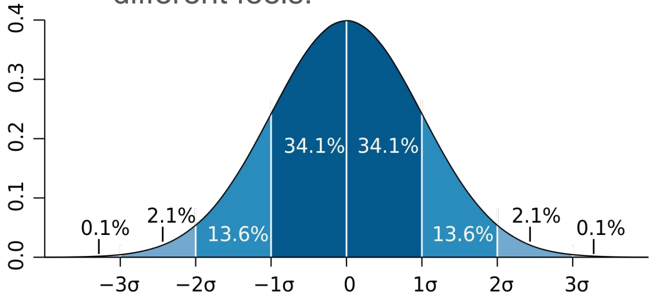
Uniform Noise

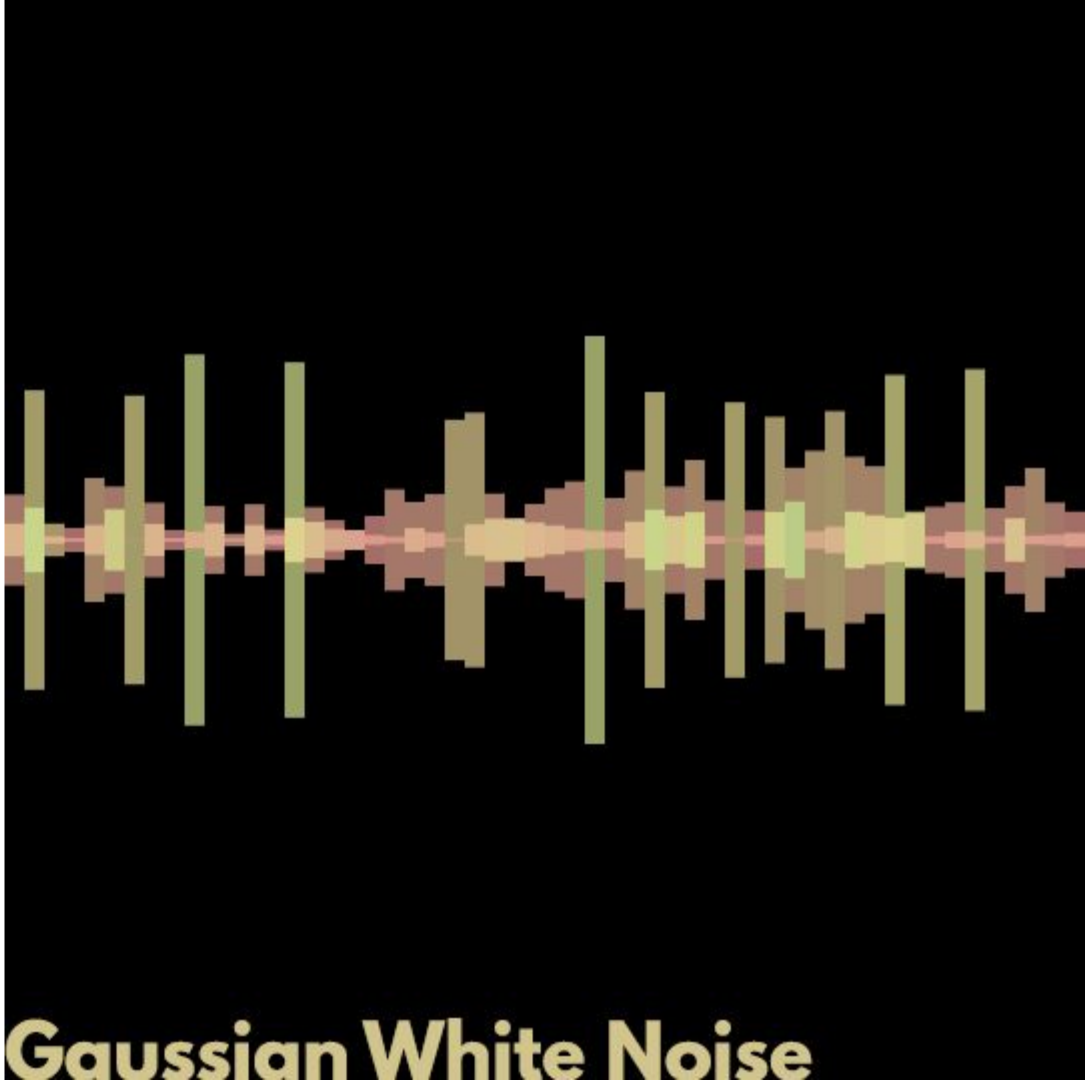
Here’s a different way at looking at the same generator.



Distribution

- You don't need to limit yourself to an even distribution of random numbers,
- A normal/gaussian bell curve often gives a better feel than white noise.
- Other distributions can give different feels!





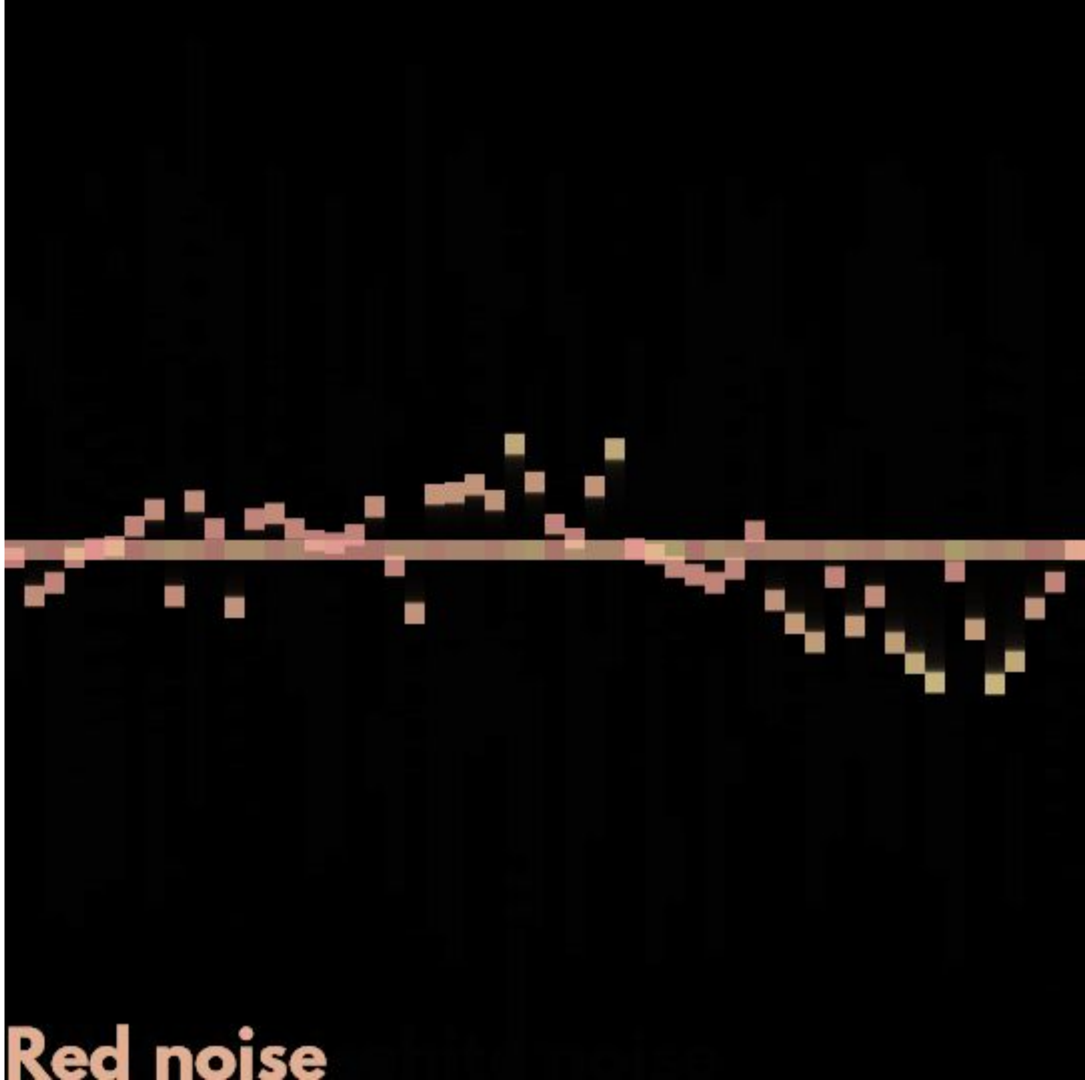
Red Noise

There are other “colors” of noise:

In red noise lower frequencies have a higher amplitude. (It is analogous to the random walk algorithm)

In violet noise, high frequencies have higher amplitude.

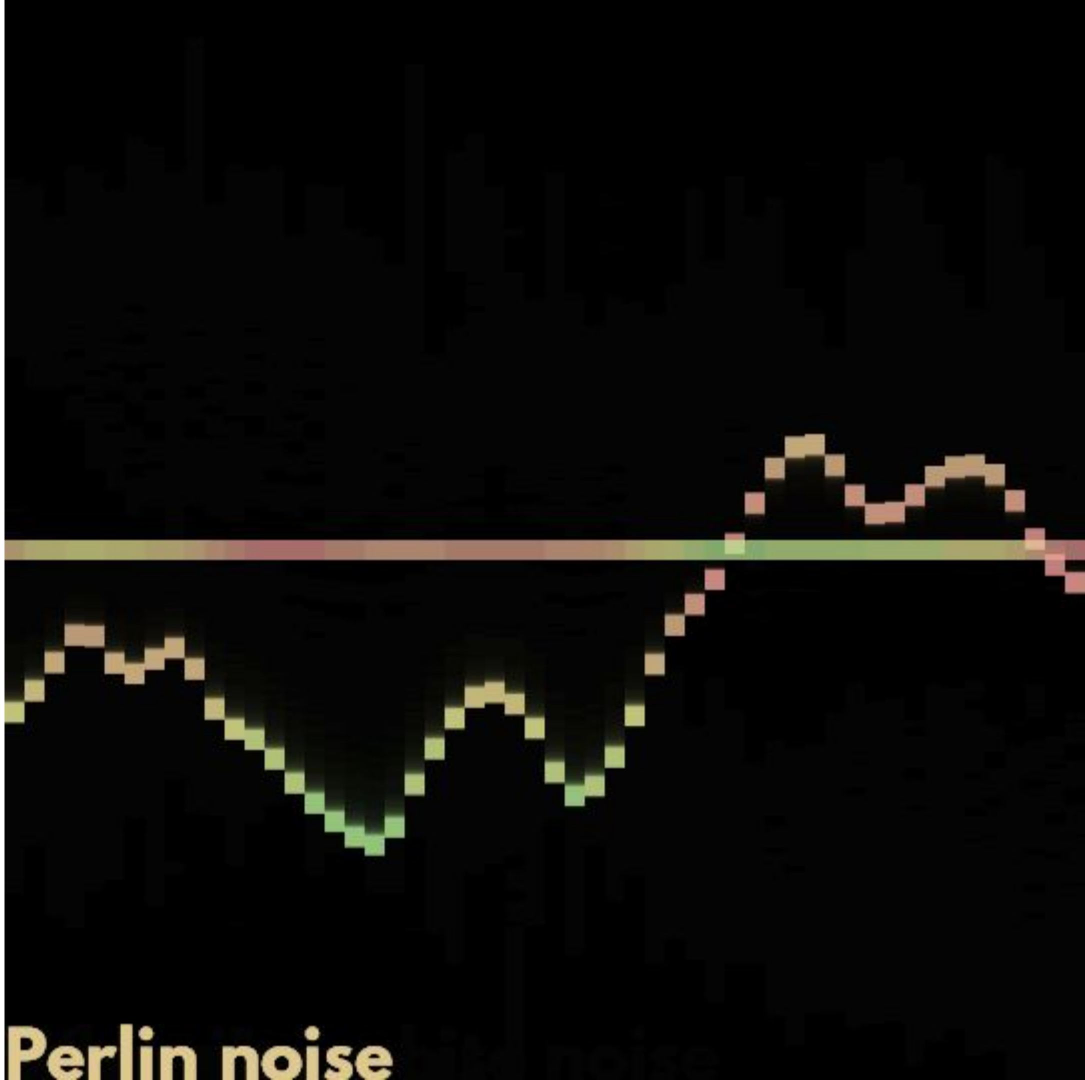
Blue noise is somewhere between violet noise and white noise, and tends to give a roughly even distribution. Blue noise often gets used for dithering.



Perlin Noise

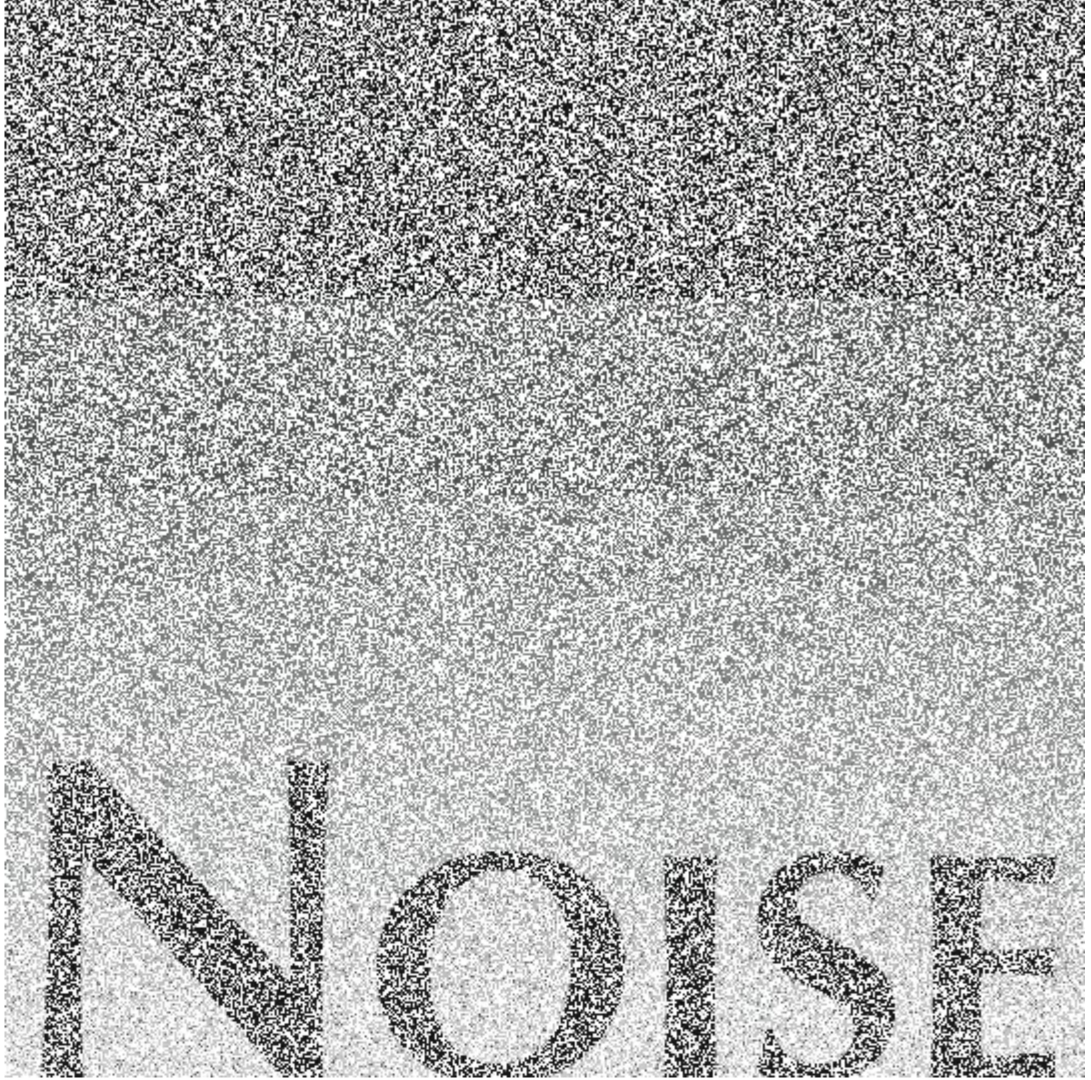
Unlike the noise we've looked at so far, the points generated by Perlin noise are related to their neighbors.

```
double noise1(double arg)
{
        int bx0, bx1;
        float rx0, rx1, sx, t, u, v, vec[1];

        vec[0] = arg;
        if (start) {
                start = 0;
                init();
        }

        setup(0, bx0,bx1, rx0,rx1);

        sx = s_curve(rx0);

        u = rx0 * g1[ p[ bx0 ] ];
        v = rx1 * g1[ p[ bx1 ] ];

        return lerp(sx, u, v);
}
```



White Noise in 2D

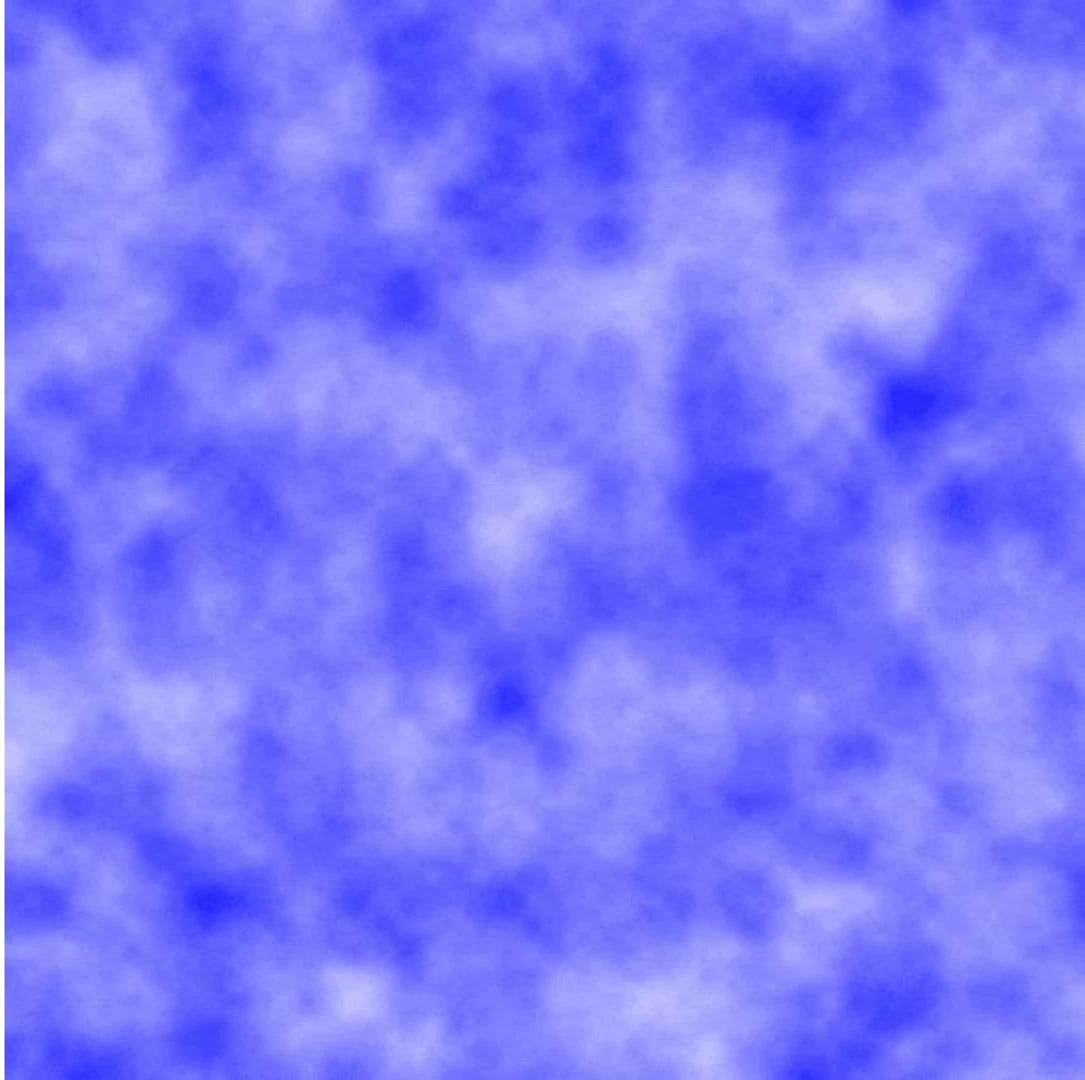
White noise in two dimensions just looks like static



Perlin Noise 2D

But Perlin noise in two dimensions start looking like clouds, or a landscape.

(Ken Perlin also invented Simplex noise, which works better in higher dimensions.)

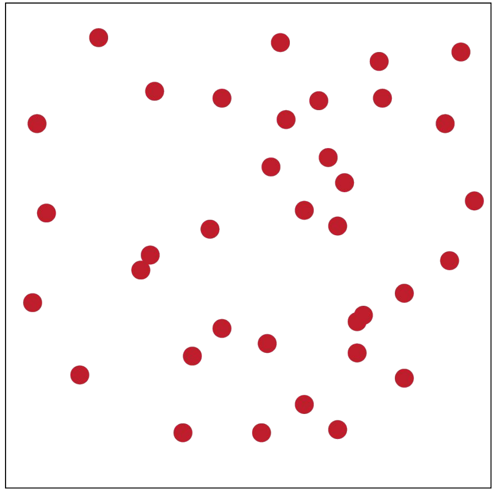


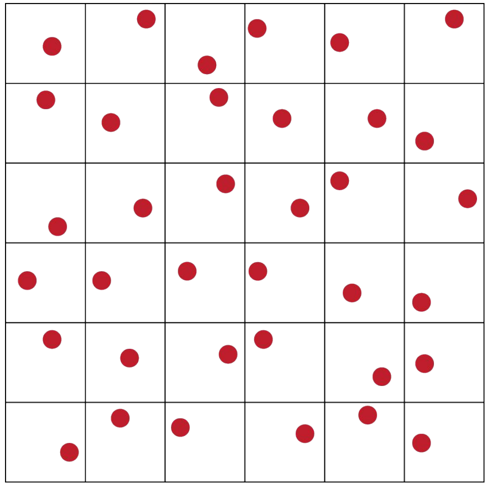
Controlling Randomness

Dice are only one algorithm for getting randomness. Sometimes other ways to distribute the data are better for the result you want.

Using playing cards (or a shuffled array) gives an entirely different kind of distribution, minimizing exact repetition. (This often fits human intuition better!)

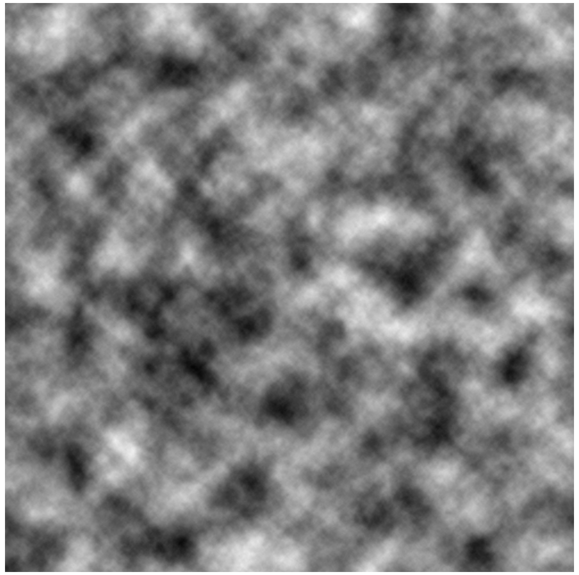
You can also do things like generating points in an offset grid. This gives a fairly even but still random generation.

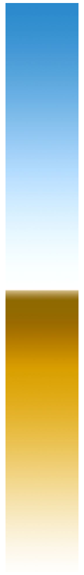


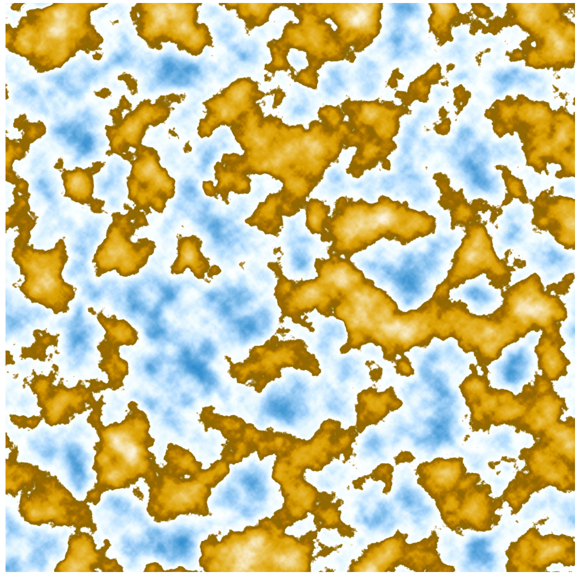


Layering Noise

We can combine noise: black and white Perlin noise on the left, times a color gradient, makes the landscape on the right.

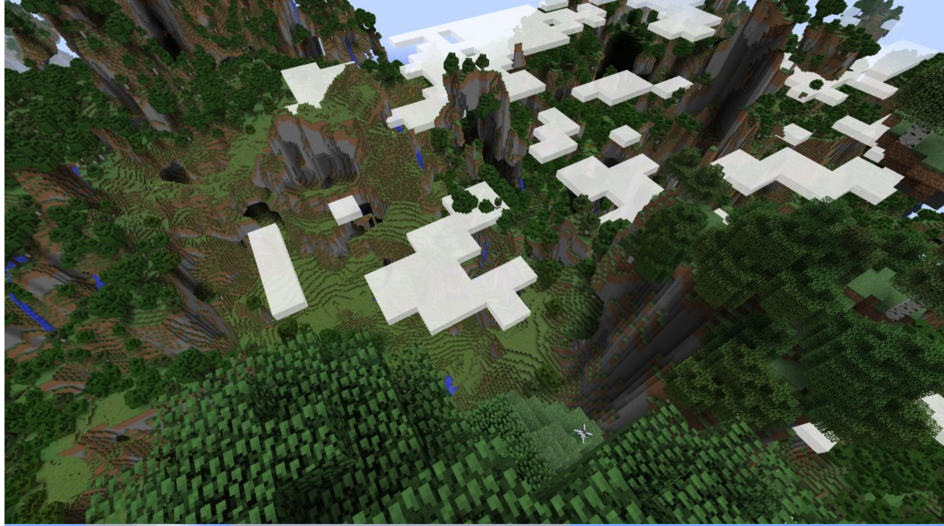
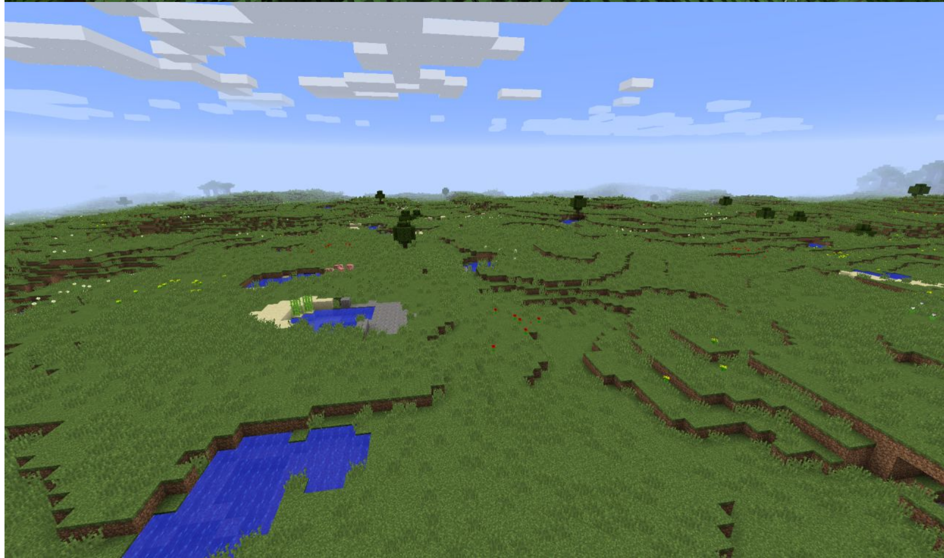






Terrain Generation

- One common use for noise is terrain generation.
- You can define a height map with Perlin noise: white becomes mountains and black becomes valleys.
- (And you can use more noise to define biomes.)
- Because you can sample each point independently, it's easy to jump to any point in an infinite world.
- A lot of games use this to make their maps, including Minecraft and No Man's Sky.

Noise Warping

Íñigo Quílez, domain warping

More Debugging Tips

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

Walk through your code step by step, explaining to yourself what is supposed to happen

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
   a. assert()
   b. Keep a debugging notebook
2. Make debug tools
   a. Quicker feedback is better
   b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
   a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
   a. Faster to print an array as a string than to individually print the contents

Useful random debugging advice

1. When you find a problem, change something so that same problem can't happen again
    a. assert()
    b. Keep a debugging notebook
2. Make debug tools
    a. Quicker feedback is better
    b. Display values live if possible
3. Only make one change at a time and then test it
4. Just because you paused the game doesn't mean it's paused
    a. And stopping one update doesn't mean you stopped all of them
5. console.log() is slow
    a. Faster to print an array as a string than to individually print the contents

AABB characters and slopes

An example of a real-world physics-and-debugging problem in a game with 2D physics like yours

https://twitter.com/eevee/status/1133248372624613376