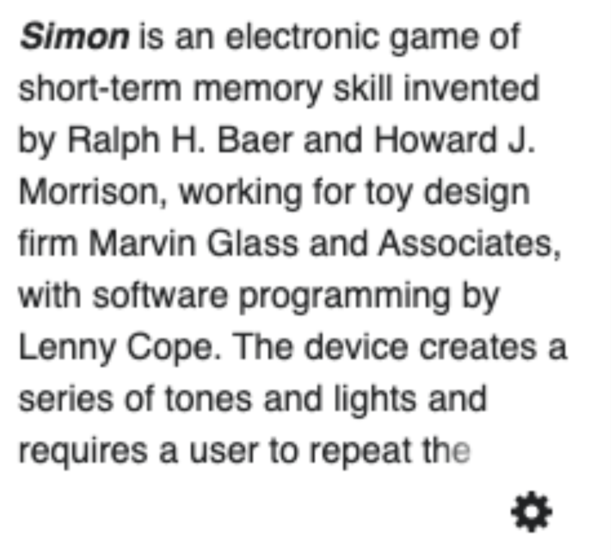
Week_21

Trinity - Introduction to Computer Programming with Game Development.

02/21/2024

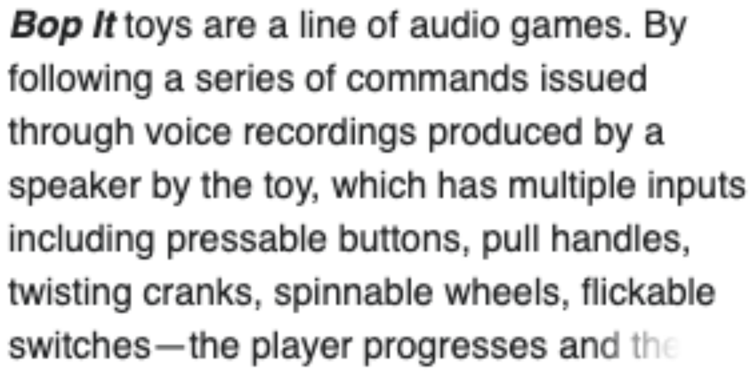
Today Lesson is about **<u>audio</u>** (Adding Sounds)

Adding the right audio, enhances the user's experience. Audio has a rich history in Game Development, dating back from the 1970's when the Milton Bradley released a hand held audio game called "Simon".

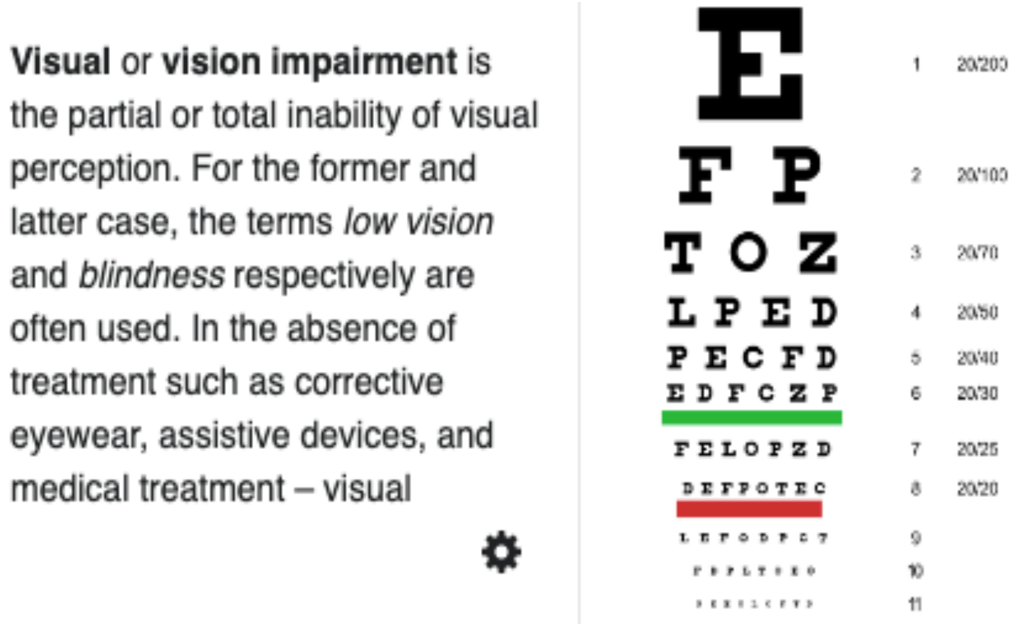


Simon is an electronic game of short-term memory skill invented by Ralph H. Baer and Howard J. Morrison, working for toy design firm Marvin Glass and Associates, with software programming by Lenny Cope. The device creates a series of tones and lights and requires a user to repeat the

Later in the 1990's the "Bop it" was released:


Bop It toys are a line of audio games. By following a series of commands issued through voice recordings produced by a speaker by the toy, which has multiple inputs including pressable buttons, pull handles, twisting cranks, spinnable wheels, flickable switches—the player progresses and the

Later advancements in audio included TTS (Text To Speech) and this is becoming more popular in current games. Another important use for audio is for the visually impaired community


Visual or **vision impairment** is the partial or total inability of visual perception. For the former and latter case, the terms *low vision* and *blindness* respectively are often used. In the absence of treatment such as corrective eyewear, assistive devices, and medical treatment – visual

Audio accessibilities are very helpful to this community!

With the rise in popularity of voice assistants such as amazon alexa came a new set of audio games. As of June 2021, 10,000 audio games were available as Alexa Skills for use with Amazon Alex, Among them are games like Rain Labs' *Animal Sounds*, which asks users to correctly identify the noises made by various animals.

Today we will work on our 4096 Game,
And add sound effects… This will be an interactive demo, where you will update your code with Coach Arthur to add sound to the game…
Please Pay Close Attention
Everything you need is in this (downloaded) file week_21_cp.pdf

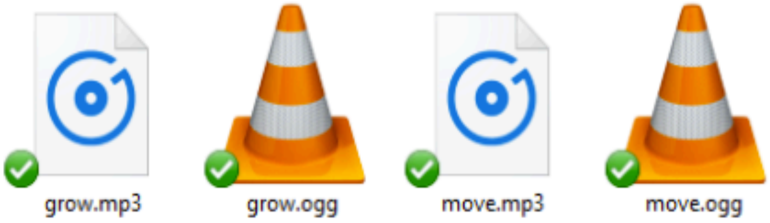
First lets review - look at Windows file Explorer / or Mac - Finder… (look at the sub-directories) - and Explore more…

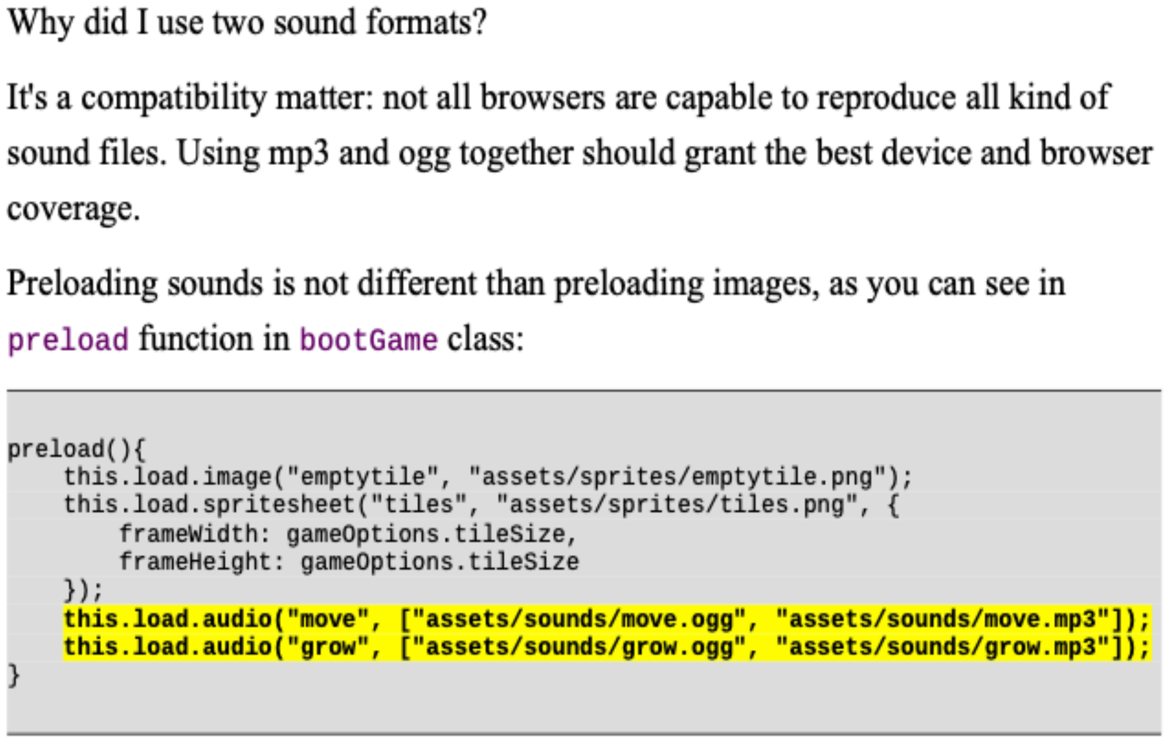
Adding Sound Effects

Our 4096 game will have two sound effects: one to be played when the player moves, and one to be played when a tile upgrades (example: [2] -> [2] => [4]).

To keep the game folder structure well organized, we are creating a new folder inside assets folder, which already contains sprites folder, and we are calling it sounds.

Two sounds are added to sounds folder, in two different formats: **mp3** and **ogg**, making the folder content look like this:





Why did I use two sound formats?

It's a compatibility matter: not all browsers are capable to reproduce all kind of sound files. Using mp3 and ogg together should grant the best device and browser coverage.

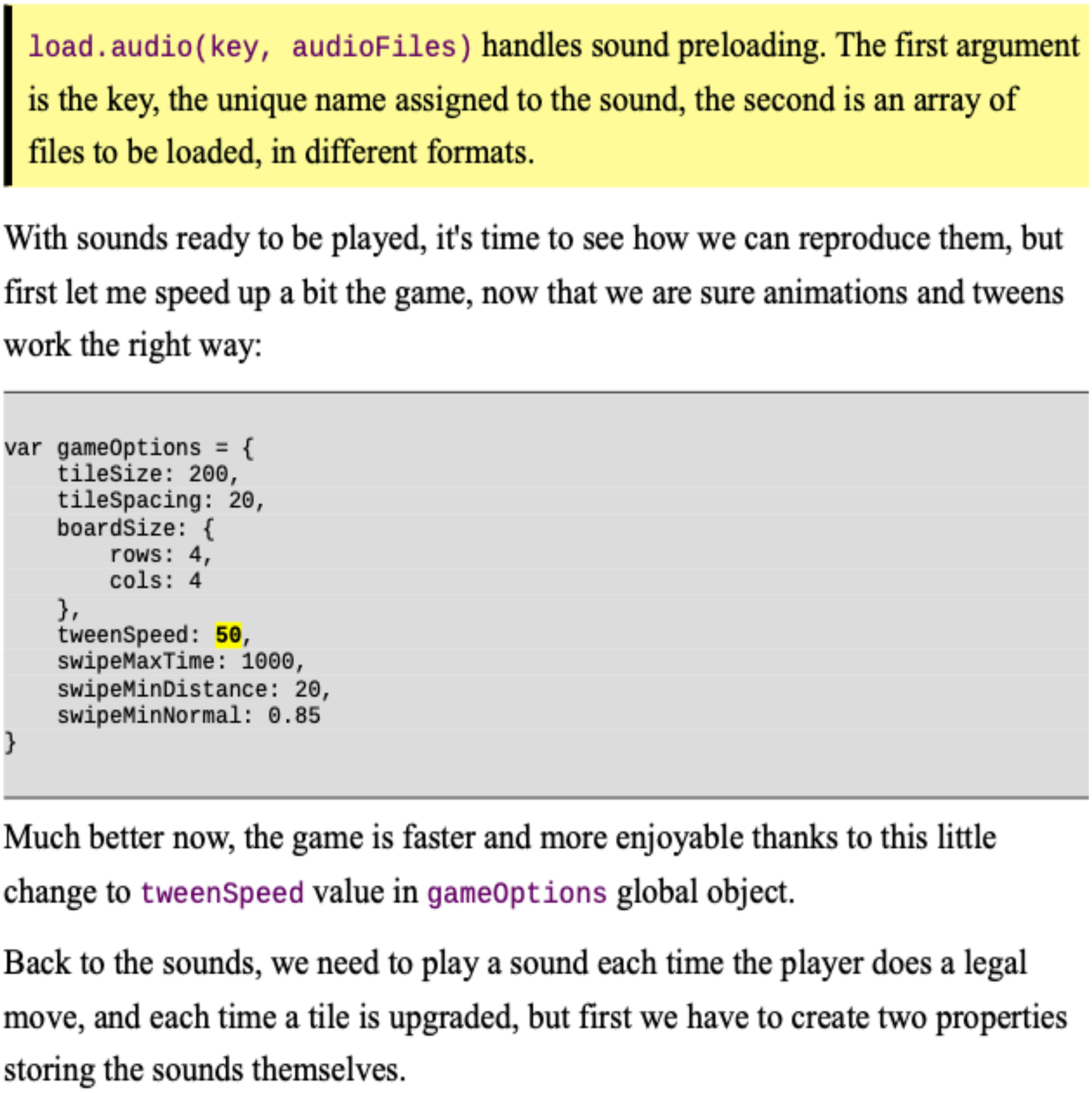
Preloading sounds is not different than preloading images, as you can see in `preload` function in `bootGame` class:

```
preload(){
    this.load.image("emptytile", "assets/sprites/emptytile.png");
    this.load.spritesheet("tiles", "assets/sprites/tiles.png", {
        frameWidth: gameOptions.tileSize,
        frameHeight: gameOptions.tileSize
    });
    this.load.audio("move", ["assets/sounds/move.ogg", "assets/sounds/move.mp3"]);
    this.load.audio("grow", ["assets/sounds/grow.ogg", "assets/sounds/grow.mp3"]);
}
```

Add these lines to the preload() function:
lines 42 & 43 of the game.js (in the downloads), line 42 already exists for you!

already exists for you:

this.load.audio("move", ["assets/sounds/move.ogg", "assets/sounds/move.mp3"]);
this.load.audio("grow", ["assets/sounds/grow.ogg", "assets/sounds/grow.mp3"]);



> `load.audio(key, audioFiles)` handles sound preloading. The first argument is the key, the unique name assigned to the sound, the second is an array of files to be loaded, in different formats.

With sounds ready to be played, it's time to see how we can reproduce them, but first let me speed up a bit the game, now that we are sure animations and tweens work the right way:

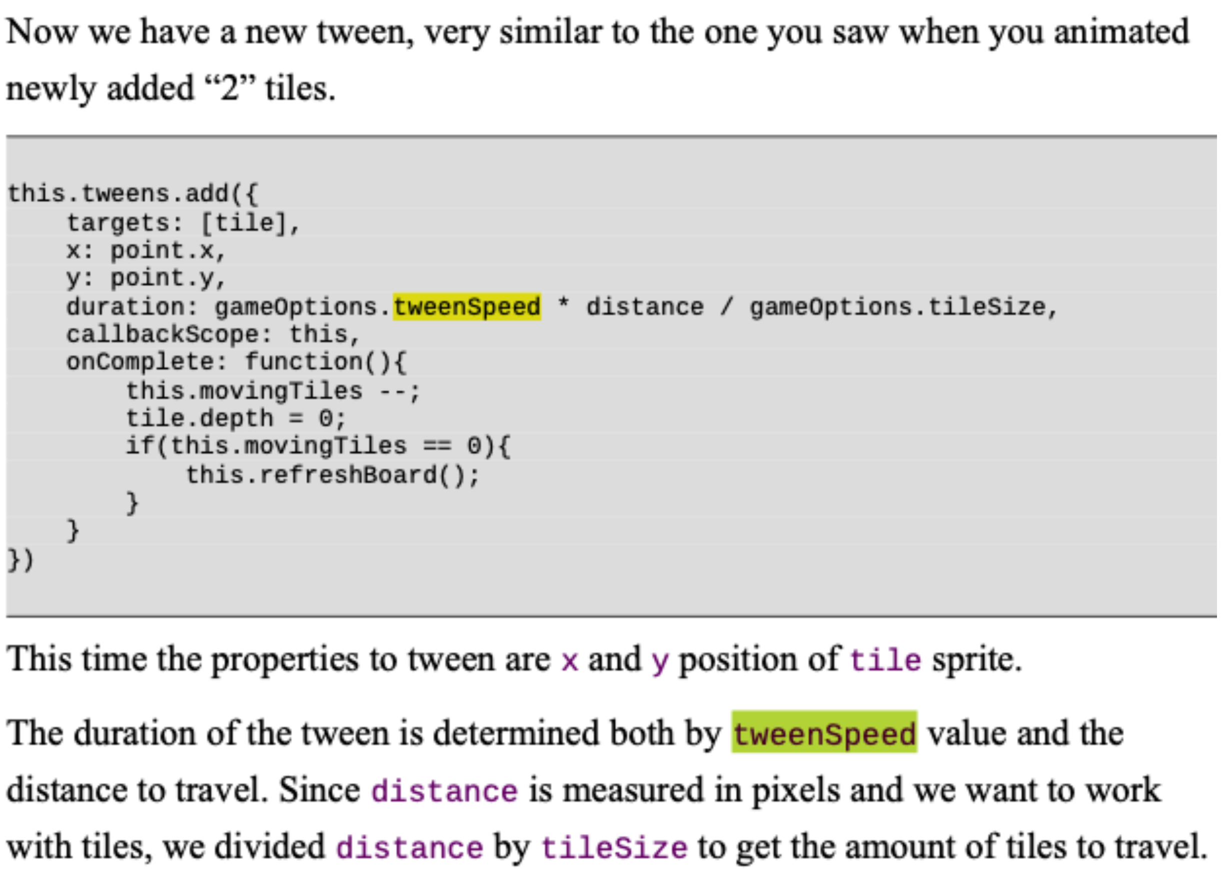
```
var gameOptions = {
    tileSize: 200,
    tileSpacing: 20,
    boardSize: {
        rows: 4,
        cols: 4
    },
    tweenSpeed: 50,
    swipeMaxTime: 1000,
    swipeMinDistance: 20,
    swipeMinNormal: 0.85
}
```

Much better now, the game is faster and more enjoyable thanks to this little change to `tweenSpeed` value in `gameOptions` global object.

Back to the sounds, we need to play a sound each time the player does a legal move, and each time a tile is upgraded, but first we have to create two properties storing the sounds themselves.

On line 9 change the tweenSpeed from 200 to 50,

Quick Review of TweenSpeed: (Line 266 - this is done)



Now we have a new tween, very similar to the one you saw when you animated newly added “2” tiles.

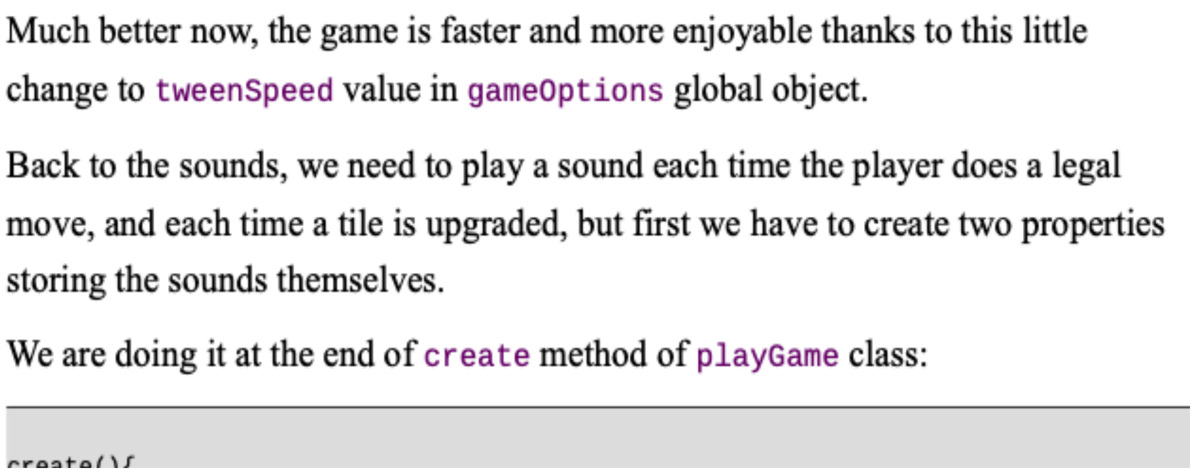
```
this.tweens.add({
    targets: [tile],
    x: point.x,
    y: point.y,
    duration: gameOptions.tweenSpeed * distance / gameOptions.tileSize,
    callbackScope: this,
    onComplete: function(){
        this.movingTiles --;
        tile.depth = 0;
        if(this.movingTiles == 0){
            this.refreshBoard();
        }
    }
})
```

This time the properties to tween are `x` and `y` position of `tile` sprite.

The duration of the tween is determined both by `tweenSpeed` value and the distance to travel. Since `distance` is measured in pixels and we want to work with tiles, we divided `distance` by `tileSize` to get the amount of tiles to travel.

Therefore the animation will happen faster with a lower value.

Next, (Create is line 54), add these at line 75 & 76

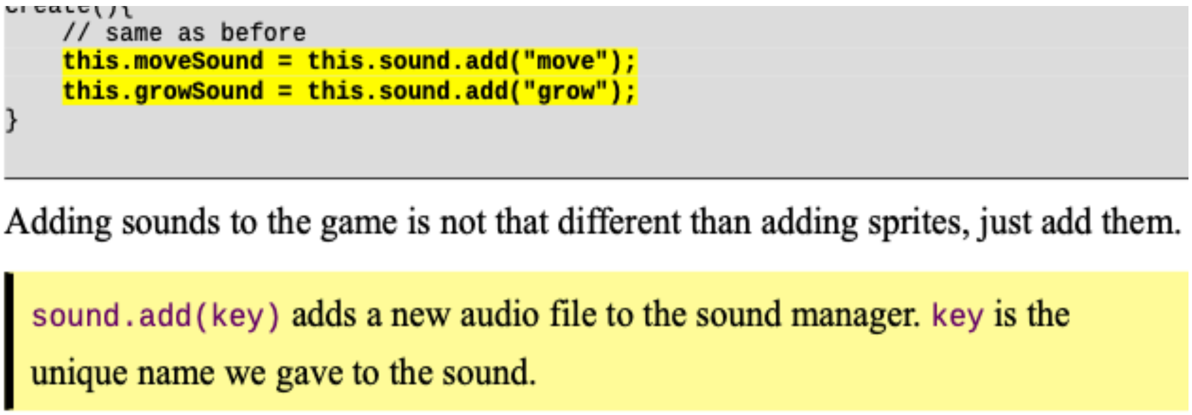


Much better now, the game is faster and more enjoyable thanks to this little change to `tweenSpeed` value in `gameOptions` global object.

Back to the sounds, we need to play a sound each time the player does a legal move, and each time a tile is upgraded, but first we have to create two properties storing the sounds themselves.

We are doing it at the end of `create` method of `playGame` class:

```
create(){
```



```
create(){
    // same as before
    this.moveSound = this.sound.add("move");
    this.growSound = this.sound.add("grow");
}
```

Adding sounds to the game is not that different than adding sprites, just add them.

`sound.add(key)` adds a new audio file to the sound manager. `key` is the unique name we gave to the sound.

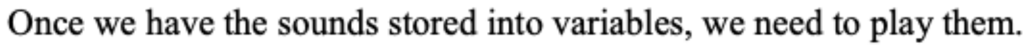
There is a class called playGame that extends the Phaser.Scene
(line 48), on line 52 begins the create method of this class;
When this class gets created…

Add these lines at 75 (un-comment by removing the //) & 76

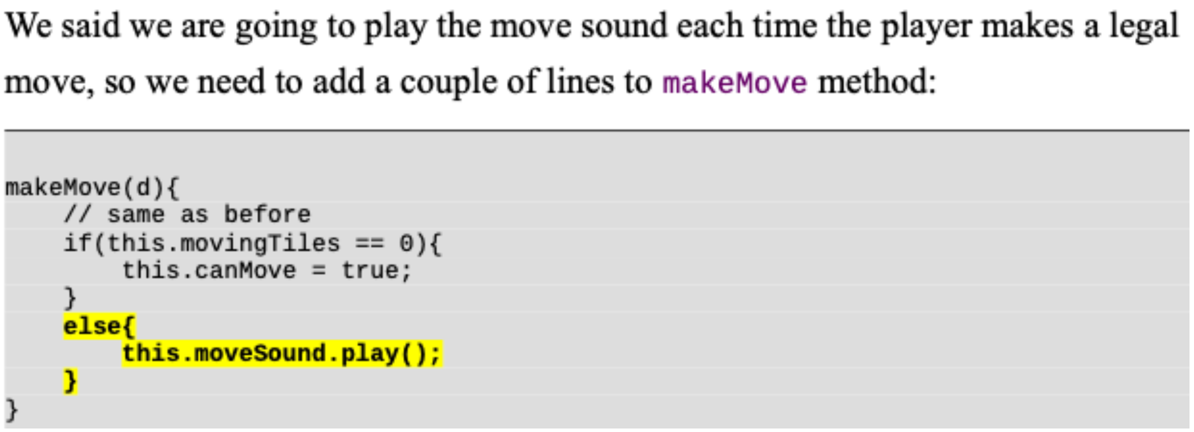
this.moveSound = this.sound.add("move");
this.growSound = this.sound.add("grow");

Next,

Once we have the sounds stored into variables, we need to play them.
We said we are going to play the move sound each time the player makes a legal
move, so we need to add a couple of lines to makeMove method:



Once we have the sounds stored into variables, we need to play them.



We said we are going to play the move sound each time the player makes a legal move, so we need to add a couple of lines to `makeMove` method:

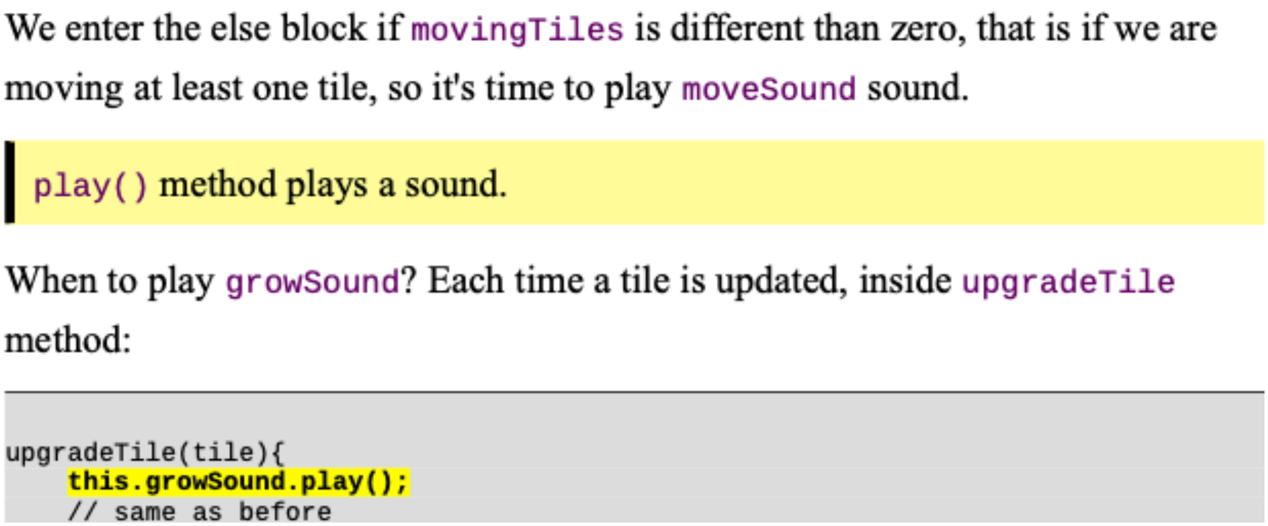
```
makeMove(d){
    // same as before
    if(this.movingTiles == 0){
        this.canMove = true;
    }
    else{
        this.moveSound.play();
    }
}
```

The MakeMove(method) begins at line 158.

At line 198 add these 3 lines:

```
else{
         this.moveSound.play();
      }
```

Next,

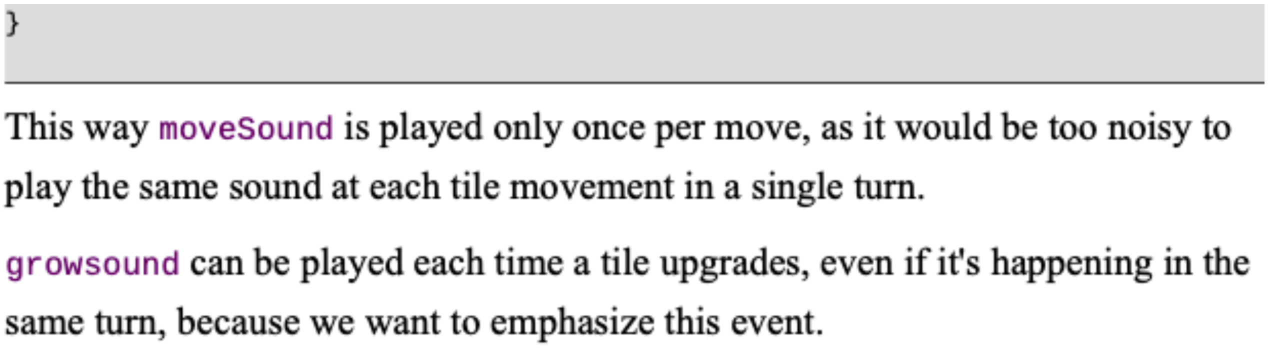


We enter the else block if `movingTiles` is different than zero, that is if we are moving at least one tile, so it's time to play `moveSound` sound.

> `play()` method plays a sound.

When to play `growSound`? Each time a tile is updated, inside `upgradeTile` method:

```
upgradeTile(tile){
    this.growSound.play();
    // same as before
```



```
}
```

This way `moveSound` is played only once per move, as it would be too noisy to play the same sound at each tile movement in a single turn.

`growsound` can be played each time a tile upgrades, even if it's happening in the same turn, because we want to emphasize this event.

The UpgradeTile(method) begins at line 221

At line 223 , remove the comment :

//this.growSound.play();
((remove the "//" which is a comment))

Great Job,

Now we will test this:
Create a new Fenix Web Server,
And point it to this (dir)
<<similar to>>
C:\Trinity\Compute_Programming\Week21\sounds