

Cat Mario / Unfair Mario

- Frustration
- Oväntade dödsfall
- Fake spikar, osynliga spikar osv.



Arbetssätt

- Gruppkontrakt (När vi träffas / hur ofta vi träffas osv.)
- Uppdelning av arbete
- Arbetsföljd
- Hjälp av gruppmedlemmar

Design och ljud

TS interfaces.ts ✕

```
✓ interface Fonts {  
  roboto: p5.Font;  
  mcLawsuit: p5.Font;  
  pressStart2p : p5.Font  
}  
  
✓ interface sound {  
  backGroundMusic: p5.SoundFile;  
  jump: p5.SoundFile;  
  deathSong: p5.SoundFile;  
}
```

TS sketch.ts ✕

```
function preload() {  
  
  /** Fonts */  
  fonts = {  
    roboto: loadFont('./assets/fonts/Roboto-Regular.ttf'),  
    mcLawsuit: loadFont('./assets/fonts/mclawsui.ttf'),  
    pressStart2p: loadFont('./assets/fonts/PressStart2P-Regular.ttf')  
  }  
  
  sound = {  
    backGroundMusic: new p5.SoundFile('./audioFiles/soundtrack.mp3'),  
    jump: new p5.SoundFile('./audiofiles/jump.wav'),  
    deathSong: new p5.SoundFile('./audiofiles/game-over.mp3')  
  }  
}
```



Ronald



Player + design // Love

TS interfaces.ts ✕

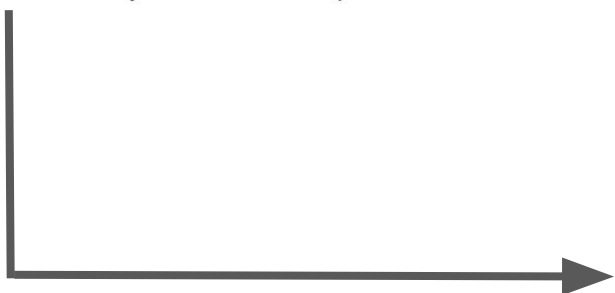
```
31 interface Sequences {  
32   idle: p5.Image[];  
33   walkLeft: p5.Image[];  
34   walkRight: p5.Image[];  
35   jumpLeft: p5.Image[];  
36   jumpRight: p5.Image[];  
37   die: p5.Image[];  
38 }
```



```
63 sequences = {  
64  
65   idle: [  
66     loadImage('./assets/images/ronald/idle-left/0.png'),  
67     loadImage('./assets/images/ronald/idle-left/1.png'),  
68     loadImage('./assets/images/ronald/idle-left/2.png'),  
69     loadImage('./assets/images/ronald/idle-left/3.png'),  
70     loadImage('./assets/images/ronald/idle-left/4.png'),  
71     loadImage('./assets/images/ronald/idle-left/5.png'),  
72     loadImage('./assets/images/ronald/idle-left/6.png'),  
73     loadImage('./assets/images/ronald/idle-left/7.png'),  
74     loadImage('./assets/images/ronald/idle-left/8.png')  
75   ],  
76  
77   walkLeft: [  
78     loadImage('./assets/images/ronald/walk-left/0.png'),  
79     loadImage('./assets/images/ronald/walk-left/1.png'),  
80     loadImage('./assets/images/ronald/walk-left/2.png'),  
81     loadImage('./assets/images/ronald/walk-left/3.png'),  
82     loadImage('./assets/images/ronald/walk-left/4.png'),  
83     loadImage('./assets/images/ronald/walk-left/5.png')  
84   ],  
85  
86   walkRight: [  
87     loadImage('./assets/images/ronald/walk-right/0.png'),  
88     loadImage('./assets/images/ronald/walk-right/1.png'),  
89     loadImage('./assets/images/ronald/walk-right/2.png'),  
90     loadImage('./assets/images/ronald/walk-right/3.png'),  
91     loadImage('./assets/images/ronald/walk-right/4.png'),  
92     loadImage('./assets/images/ronald/walk-right/5.png')  
93   ],  
94 }
```

Från stillbilder till bildspel (GIF)

```
-  
this.timeToChangeFrame = 100;  
this.sequenceIndex = 0;  
this.activeSequence = sequences.idle;
```



```
else if (!this.isOnGround) {  
    if (this.velocity.x <= 0) {  
        this.activeSequence = sequences.jumpLeft  
    }  
    else { this.activeSequence = sequences.jumpRight }  
}  
  
else if (keyIsDown(76)) { // L  
    this.activeSequence = sequences.walkRight;  
}  
  
else if (keyIsDown(65)) { // A  
    this.activeSequence = sequences.walkLeft  
}  
  
else {  
    this.activeSequence = sequences.idle;  
}
```


Enemies

```
handleCollision(entity: Entity, direction: string): void {
    super.handleCollision(entity, direction)
    if (this.direction == "horizontal") {
        switch (direction) {
            case 'left':
                this.velocity.x = Math.abs(this.velocity.x)
                // this.acceleration.x = Math.abs(this.acceleration.x)
                break;
            case 'right':
                this.velocity.x = -Math.abs(this.velocity.x)
                // this.acceleration.x = -Math.abs(this.acceleration.x)
                break;
        }
    } else {
        switch (direction) {
            case 'top':
                this.velocity.y = Math.abs(this.velocity.y)
                this.acceleration.y = Math.abs(this.acceleration.y)
                break;
            case 'bottom':
                this.velocity.y = -Math.abs(this.velocity.y)
                this.acceleration.y = -Math.abs(this.acceleration.y)
                break;
        }
    }
}
```

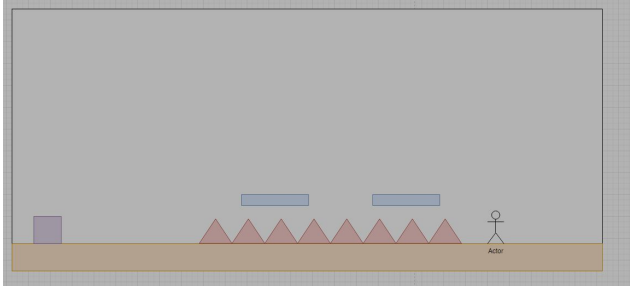
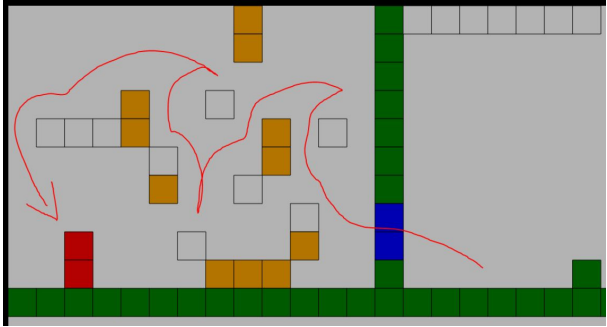
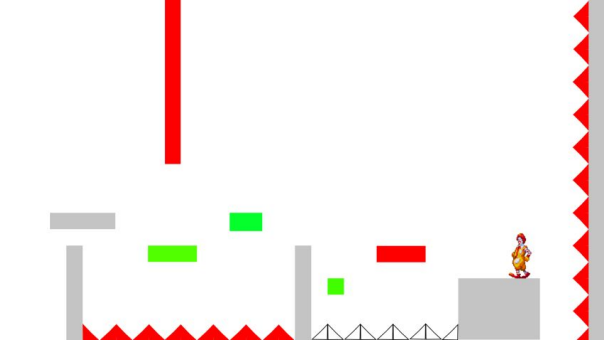
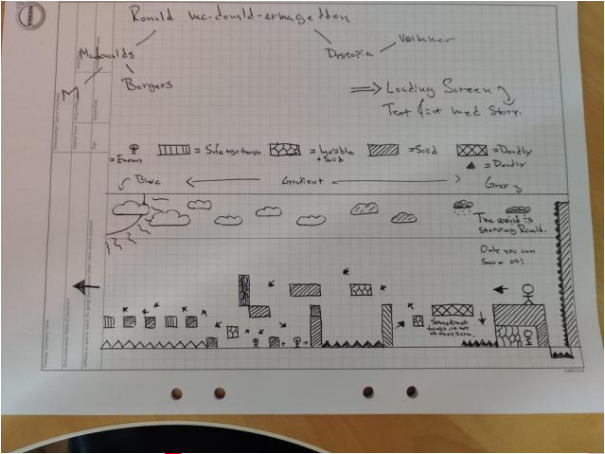
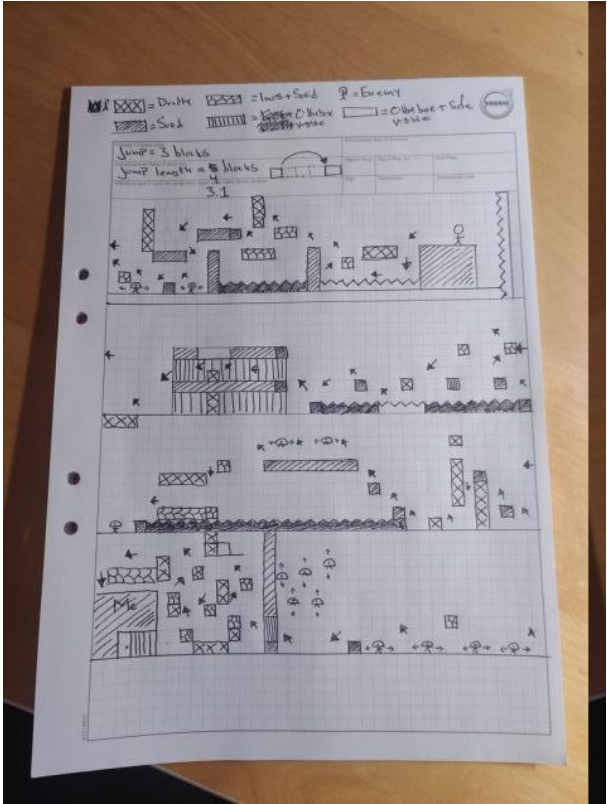
```
update(): void {
    if (this.direction == "vertical") {
        if (this.position.y <= 0) {
            this.velocity.y = 4
            this.acceleration.y = 0
        }
        if (this.position.y >= height - this.size.y) {
            this.velocity.y = -4
            this.acceleration.y = 0
        }
    }
    super.update()
}
```



Idégenerering



Level Design

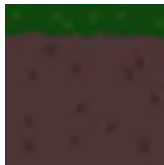
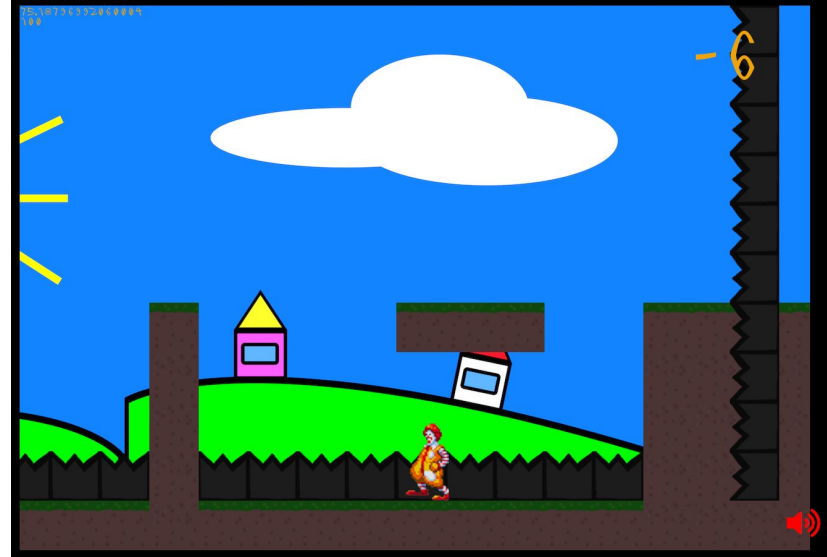


Level design & LevelGenerator

```
100, seconds ago [4 authors (100 and others)]
class Generator {
  private worldGrid: number[][];

  constructor() {

    // 1 = spik + deadly
    // 2 = spik + safe
    // 3 = solid + safe + grass
    // 4 = invis + solid + safe
    // 5 = invis + deadly
    // 6 = visible + deadly
    // 7 = visible + safe + walk through + dirt
    // 8 = vertical enemy
    // 9 = horizontal enemy
    // 10 = complete game block
    // 11 = solid + safe + dirt
    // 12 = grass + no hitbox
    // 13 = deadly + dirt
    // 14 = vertical spike + deadly
    // 15 = final top block
    // 16 = final bottom block
    // 17 = checkpoint
  }
}
```




```
public getNextLevelEntities(): Set<Entity> {  
    //1 kollar på vår grid  
    //2 och skapar alla entiter som behövs  
  
    const entities = new Set<Entity>();  
    const blockNums = new Set<number>([1, 3, 4, 5, 6, 11, 13]);  
  
    for (let y in this.worldGrid) {  
        for (let x in this.worldGrid[y]) {  
            // x*80 = x  
            // y*80 = y  
            const index = createVector(parseInt(x), parseInt(y));  
            const position = p5.Vector.mult(index, 80);  
  
            switch (this.worldGrid[y][x]) {  
                case 1 /* new x(position, fill, isSolid, damage,)*/*:  
                    entities.add(new Spike(position, images.spikeBlock, true, true));  
                    break;  
  
                case 2:  
                    entities.add(new Spike(position, images.spikeBlock, false, false));  
                    break;  
  
                case 3:  
                    entities.add(new Block(position, images.grassBlock, true, false, Tools.neighborsFree(this.worldGrid, index, blockNums)));  
                    break;  
  
                case 4:  
                    entities.add(new Block(position, images.invisBlock, true, false, Tools.neighborsFree(this.worldGrid, index, blockNums)));  
                    break;  
  
                case 5:  
                    entities.add(new Block(position, images.invisBlock, true, true));  
                    break;  
            }  
        }  
    }  
}
```


Collisions

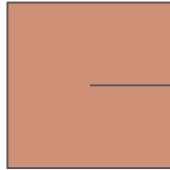
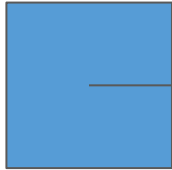
```
private detectCollisions() {  
    for (const e0 of this.collidableEntities.animated) {  
        for (const e1 of this.collidableEntities.static) {  
            this.detectCollision(e0, e1);  
        }  
    }  
  
    const animatedEntities = Array.from(this.collidableEntities.animated);  
    while (animatedEntities.length >= 2) {  
        const e0 = animatedEntities.pop(); //Remove from entities so it won't be checked more than once  
        //Apparently TS needs this in order to not freak out  
        if (e0?.position === undefined) throw new ReferenceError('Undefined entity position. You\'ve screwed up pretty bad.');
```

```
        for (const e1 of animatedEntities) {  
            this.detectCollision(e0, e1);  
        }  
    }  
}
```

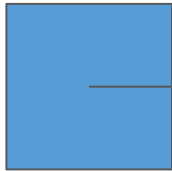
```
let relVel = p5.Vector.sub(e0.getVelocity(), e1.getVelocity());
```




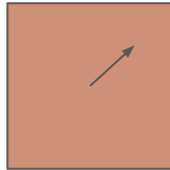
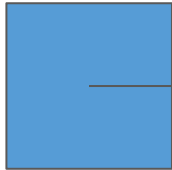
relVel:



relVel:



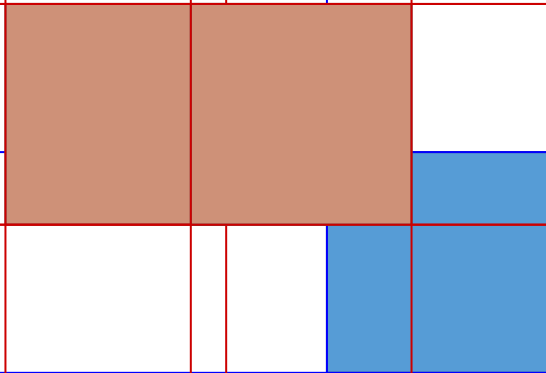
relVel: 



relVel: 


```
const overlap = {
  x: Tools.overlap([box0.left, box0.right], [box1.left, box1.right]),
  y: Tools.overlap([box0.top, box0.bottom], [box1.top, box1.bottom])
};

left: e0.position.x,
if (overlap.x && overlap.y) {...} .x,
  bottom: e0.position.y + e0.size.y
};
```



```
let backtrackFactor = p5.Vector.sub(edges0,  
edges1);  
backtrackFactor.x /= relVel.x;  
backtrackFactor.y /= relVel.y;  
let direction0 = 'none';  
let xDir0 = relVel.x > 0 ? 'right' : 'left';  
let yDir0 = relVel.y > 0 ? 'bottom' : 'top';
```

```
if (abs(backtrackFactor.x) < min(abs(backtrackFactor.y), 1.1)) {  
  direction0 = xDir0;  
} else if (abs(backtrackFactor.y) < 1.1) {  
  direction0 = yDir0;  
}  
e0.handleCollision(e1, direction0);
```



Meny klassen

```
1 //Menu class used to create the start menu and game over menu.
2 class Menu implements Visual {
3     //insert parameters necessary for both menus
4     protected message: string;
5
6     constructor(message: string) {
7         this.message = message;
8     }
9
10    update(): void {
11        if (keyIsDown(ENTER)) {
12            game.setState(new GameEngine());
13            sound.backGroundMusic.loop();
14            sound.deathSong.stop();
15        }
16    }
17
18    draw(): void {
19        // Blinking text
20        if (frameCount % 100 < 30) {
21            fill(241, 163, 10, 0)
22        } else {
23            fill(241, 163, 10)
24            textSize(40)
25            text(`${this.message}`, 640, 460);
26        }
27    }
28 }
```



Game over

```
58 class GameOverMenu extends Menu {
59     private deathBalloony = 600;
60     private deathBalloonyx = 50;
61
62     draw(): void {
63
64         //Menu text
65         image(images.hell, 0, 0)
66         image(images.graveyard, 0, 200)
67         image(images.ronaldDead, 450, 400)
68         image(images.deathBalloony, this.deathBalloonyx, this.
        deathBalloony);
```

```
85     update(): void {
86
87         this.deathBalloony -= 2;
88         this.deathBalloonyx = 30 * Math.sin(this.deathBalloony * 0.03) +
            100;
89         super.update();
90     }
91 }
```



Win Menu

```
93 class WinMenu extends Menu {
94     private textScrolly = 1100;
95     private timer = 0;
96
97
98     private Balloonx = 50;
99     private Balloony = 900;
100
101
102     private Ronaldx = 0;
103     private Ronaldy = 300;
```



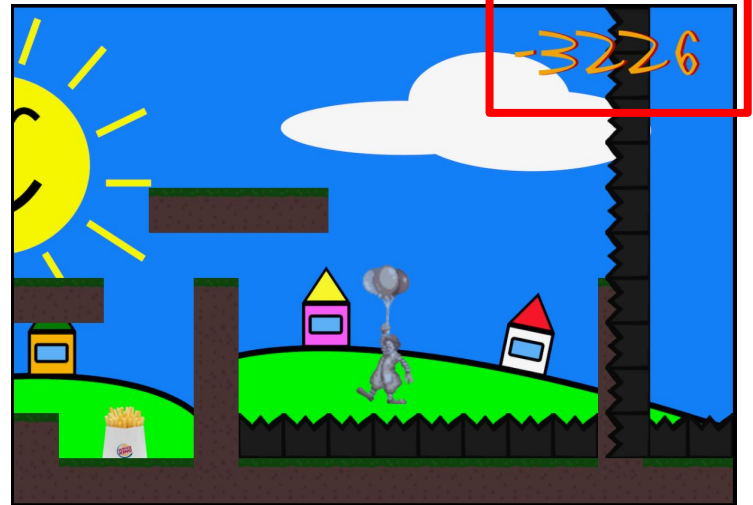
```
187 //play again text
188 if (this.timer >= 1970) {
189     textSize(40)
190     fill(241, 163, 10);
191     text('Congratulations...', 640, 350);
192     if (frameCount % 100 < 30) {
193         fill(241, 163, 10, 0);
194     } else {
195         fill(241, 163, 10);
196         text(`${this.message}`, 640, 465);
197     }
198 } else {
199     textSize(40)
200     text('Congratulations...', 640, this.textScrolly + 2200);
201 }
202
203
204 update(): void {
205     this.timer++
206     if (this.timer >= 1770) {
207         this.Ronaldy += 5
208     }
209     sound.backGroundMusic.stop();
210     document.getElementById('volumeBtn').style.display = 'none';
211     this.textScrolly -= 1.5;
212     this.Balloony -= 2;
213     this.Balloonx = 30 * Math.sin(this.Balloony * 0.03) + 100;
214     if (this.Balloony < -600) {
215         this.Balloony = 900
216     }
217     super.update();
218 }
219 }
```

DeathCounter

```
let deathCounter: DeathCounter;  
let nrOfLives = parseInt(localStorage.getItem('lives') || '3');
```

```
3 class DeathCounter implements Visual {  
4   livesNumber : number;  
5   constructor (livesNumber : number) {  
6     this.livesNumber = livesNumber  
7   }  
8   update(): void {}  
9  
10  draw(): void {  
11    textAlign(RIGHT)  
12    textSize(100)  
13    textFont(fonts.mclawsuit)  
14    fill(184, 6, 0)  
15    text(`${this.livesNumber}`, 1240, 120)  
16    fill(241, 163, 10)  
17    text(`${this.livesNumber}`, 1235, 120)  
18  }  
19 }
```

```
91 //DEAAAAAATHCOUNTER  
92 deathCounter = new DeathCounter(nrOfLives);  
93 deathCounter.draw();  
94 }
```



Reflektion

- Github issues / branches
- UML
- Grupparbete
- Discord vs Teams
- Hade vi gjort någonting annorlunda om vi hade gjort om det

Avslutningsvis

Tävling

Från kl 16:00 idag till Fredag kl 16:00

300 kr

