GruppE

Adam
Claudia
Emma
Isabel
Natalia

HOP TO THE TOP

Spelidé

Spelet går ut på att man styr en boll som ska hoppa sig igenom spelet och ta så många poäng som möjligt utan att ramla ner.

Substantiv inför klassdiagram

- Player
- Block
- GameBoard
- Level
- Item
- Game Controller
- Start screen
- GameBoard
- New highscore-animation
- How to play
- End-screen
- pause/start button

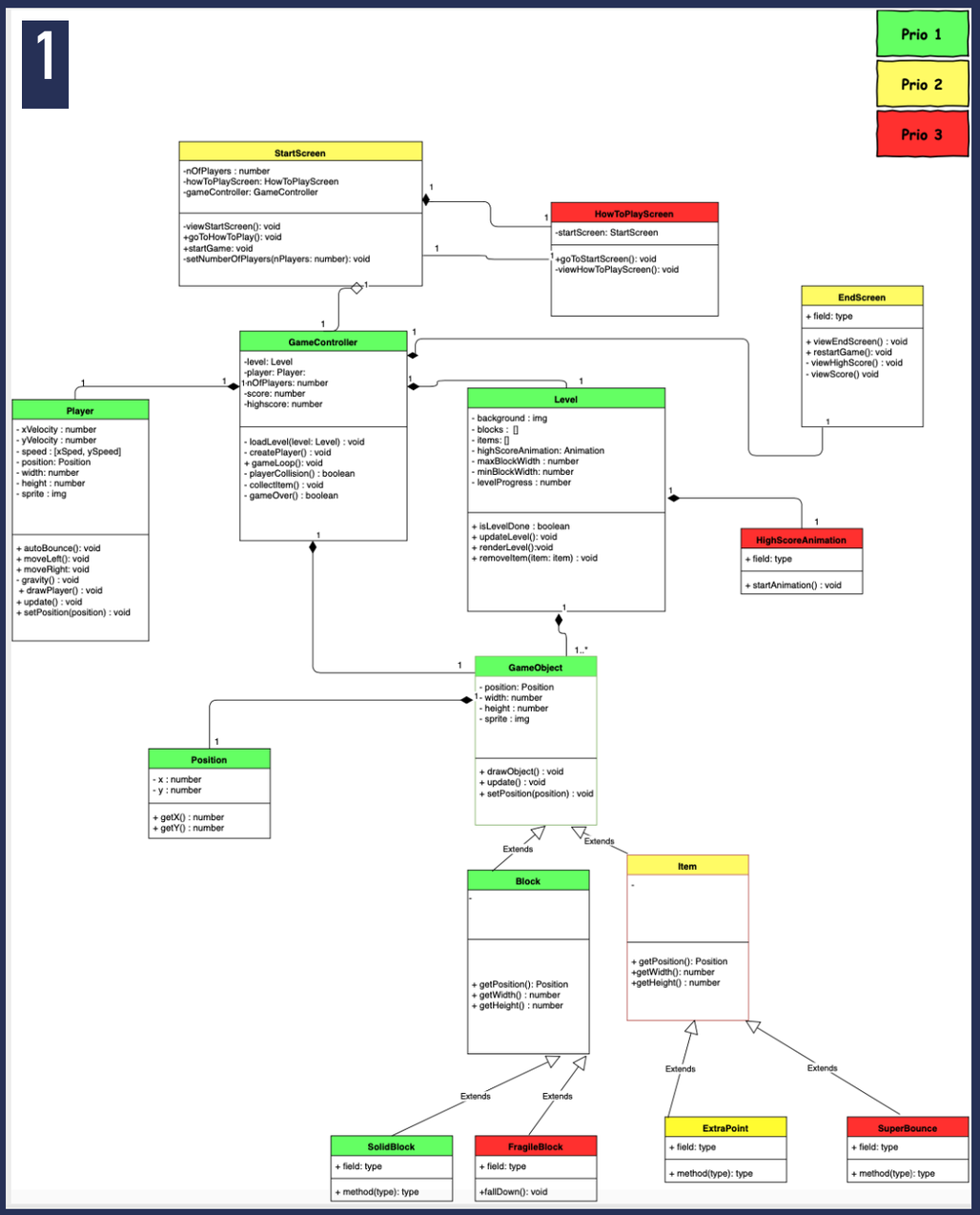
Gantt

																										Symbols:	
Month	December																	January								Adam	A
Week	50					51					52					1					2					Emma	E
Date	9	10	11	12	13	16	17	18	19	20	23	24	25	26	27	30	31	1	2	3	6	7	8	9		Claudia	C
																										Isabel	I
																										Natalia	N
planning: UML, Gantt, text-plan, group contract, sketches, github rep.	all	all	all	all	all	all	100% complete																			holidays	
Make the structure					all	all all	all	100% complete																		in progress	
																										complete	
Coding/programming in TS																										deadline	
canvas basic elements shapes											0% complete																
canvas/TS continued																						0% complete					
typescript: classes implementation																											
Update Gantt and UML																											
reconciliation(avstämning) milestone															all	100% complete											
idea presentation milestone						all	100% complete																				
presentation (inlämning) milestone																								all	0% complete		
internal release?																											
testing and bugfixing																											

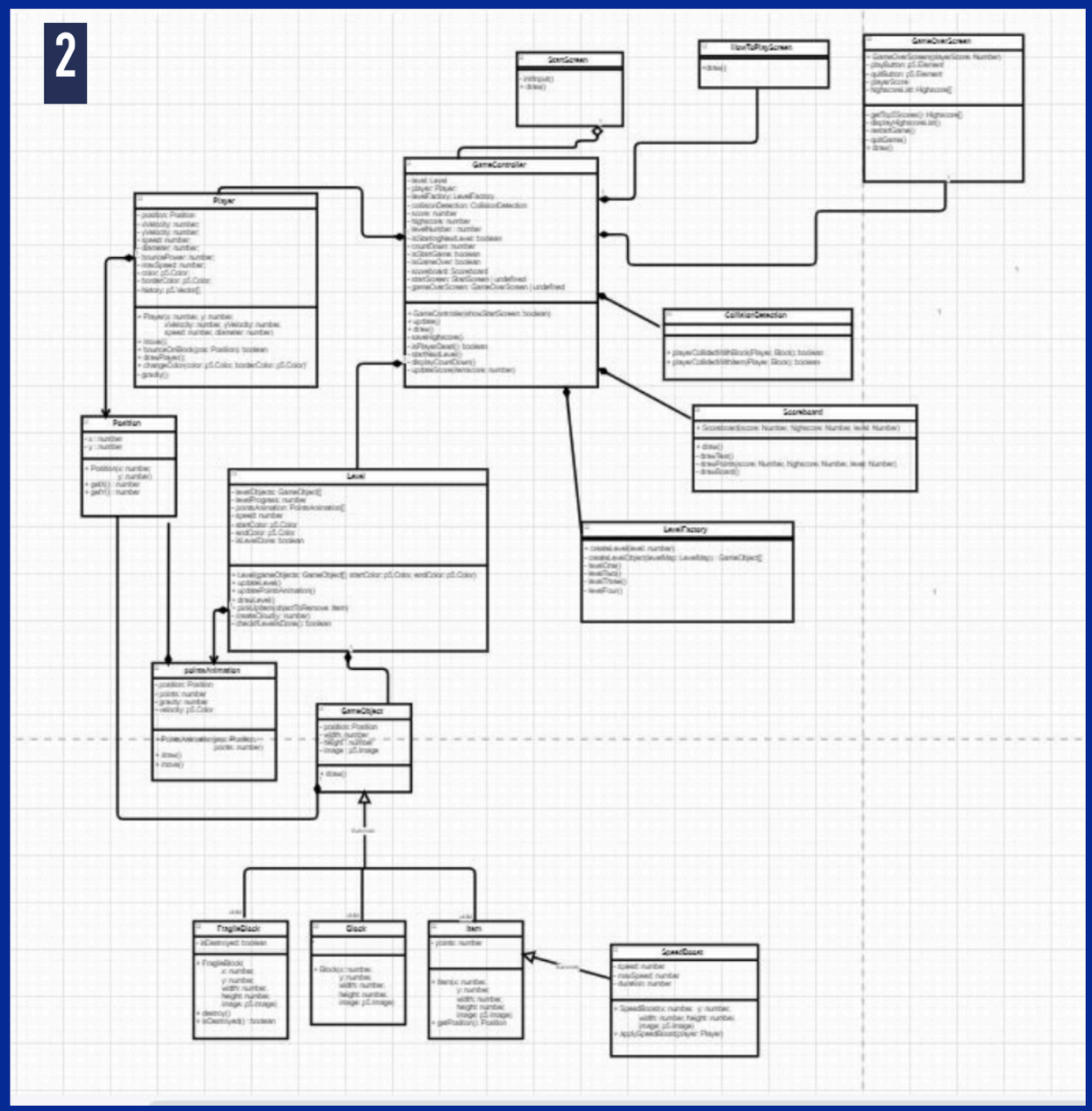
UML

1

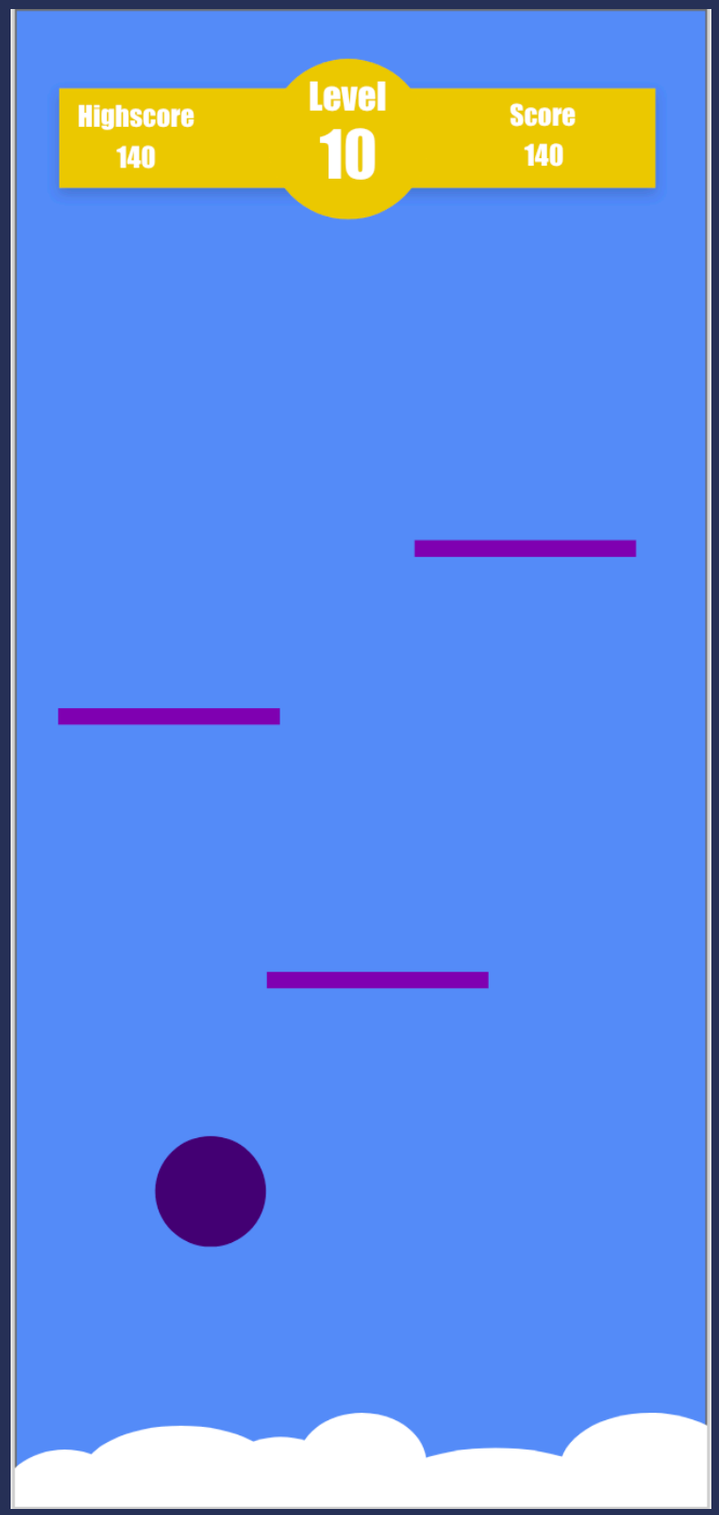
Prio 1
Prio 2
Prio 3



2



Mockup

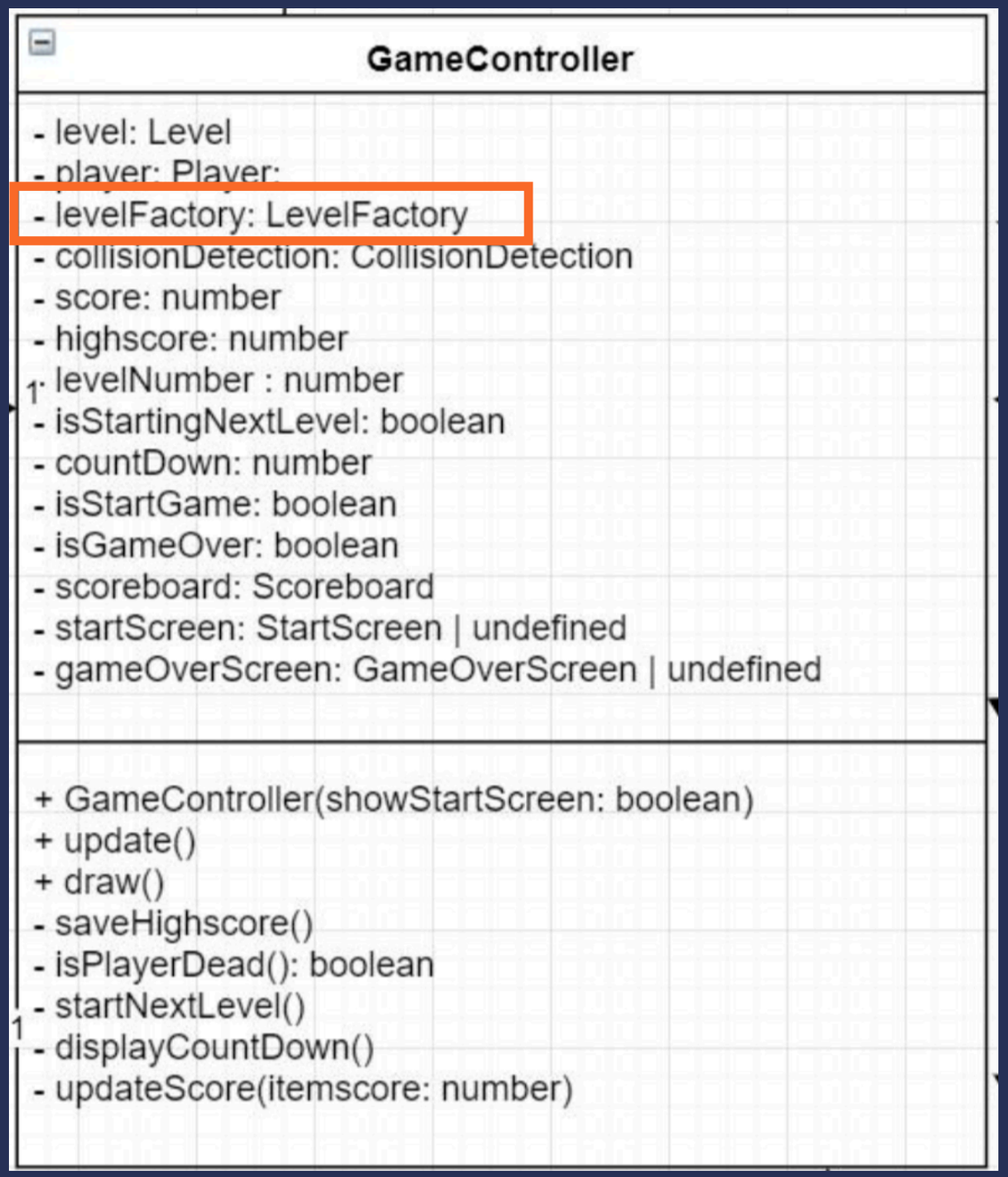


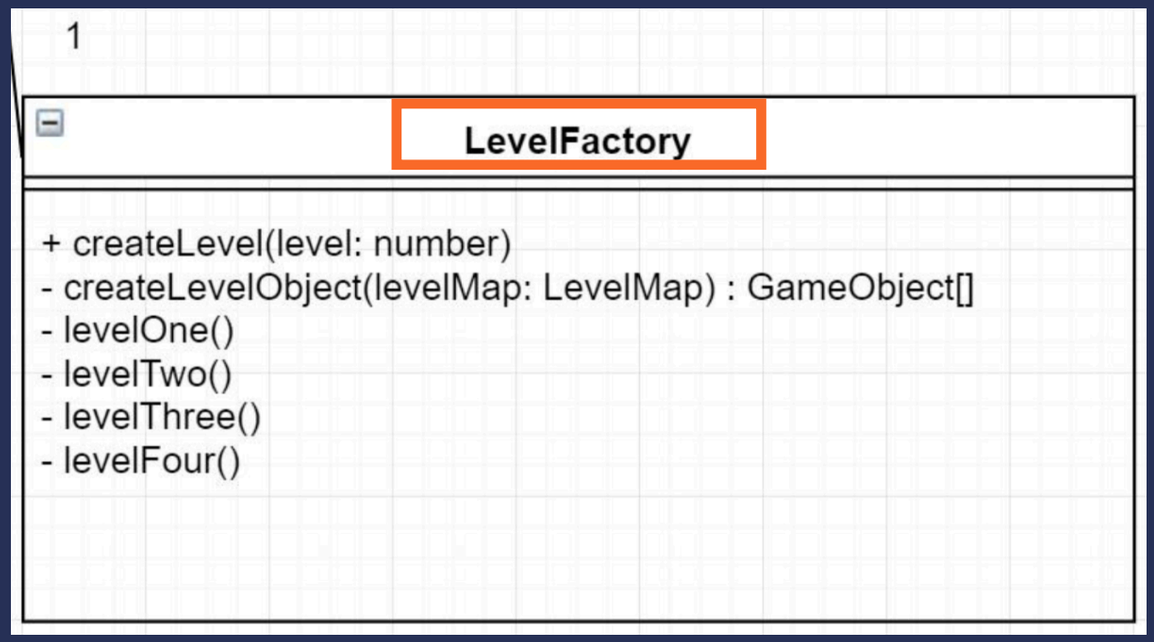
Github

Fork

Git

UML


GameController
- level: Level
- player: Player:
- levelFactory: LevelFactory
- collisionDetection: CollisionDetection
- score: number
- highscore: number
- levelNumber : number
- isStartingNextLevel: boolean
- countDown: number
- isStartGame: boolean
- isGameOver: boolean
- scoreboard: Scoreboard
- startScreen: StartScreen | undefined
- gameOverScreen: GameOverScreen | undefined
+ GameController(showStartScreen: boolean)
+ update()
+ draw()
- saveHighscore()
- isPlayerDead(): boolean
- startNextLevel()
- displayCountDown()
- updateScore(itemscore: number)
1
1


1
LevelFactory
+ createLevel(level: number)
- createLevelObject(levelMap: LevelMap) : GameObject[]
- levelOne()
- levelTwo()
- levelThree()
- levelFour()

Level factory

1

```
class LevelFactory {
  createLevel(level: number): Level {
    switch (level) {
      case 1:
        return this.levelOne();
      case 2:
        return this.levelTwo();
      case 3:
        return this.levelThree();
      case 4:
        return this.levelFour();
      case 5:
        return this.levelFive();
      default:
        return this.levelOne();
    }
  }
```

2

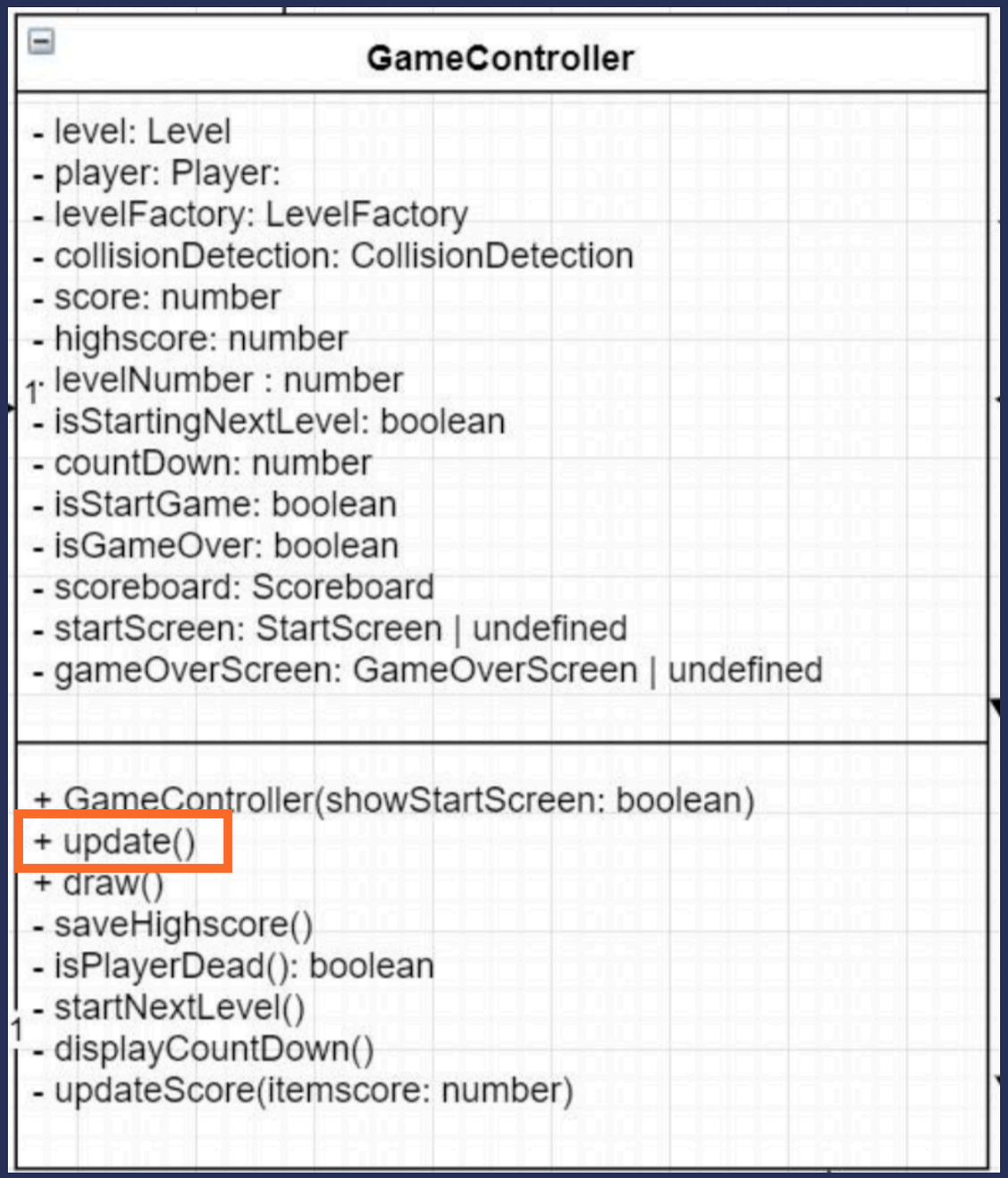
```
levelOne(): Level {
  const levelMap: LevelMap = [
    [2, 2, 2, 2, 2, 2, 2, 2],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 2, 0, 0],
    [0, 2, 2, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 6, 0],
    [0, 0, 0, 0, 0, 0, 2, 2],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 2, 2, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 2, 2],
    [0, 3, 0, 0, 0, 0, 0, 0],
    [0, 2, 2, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 2, 2, 2],
    [4, 0, 0, 0, 0, 0, 0, 0],
    [2, 2, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 6, 0],
    [0, 0, 0, 0, 2, 2, 2, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [2, 2, 2, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 2, 0, 0],
    [0, 6, 0, 0, 0, 0, 0, 0],
    [2, 2, 2, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 3],
    [0, 0, 0, 0, 0, 2, 2, 2],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [2, 2, 2, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 2, 2, 0, 0],
    [0, 5, 0, 0, 0, 0, 0, 0],
    [2, 2, 2, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 2, 2, 0, 0, 0],
    [3, 0, 0, 0, 0, 0, 0, 0],
    [2, 2, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 2, 2, 0, 0],
    [6, 0, 0, 0, 0, 0, 0, 0],
    [2, 2, 1, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [2, 2, 2, 2, 2, 2, 2, 2]
  ];
  const gameObjects: GameObject[] = this.createLevelObject(levelMap);
  const startColor = color(120, 170, 235);
  const endColor = color(50, 120, 220);
  return new Level(gameObjects, startColor, endColor);
}
```

3

```
createLevelObject(levelMap: LevelMap): GameObject[] {
  const levelObjects: GameObject[] = [];
  const xStepSize: number = width / levelMap[0].length;

  for (let y = 0; y < levelMap.length; y++) {
    for (let x = 0; x < levelMap[0].length; x++) {
      const cell = levelMap[levelMap.length - 1 - y][x];
      const xPos = x * xStepSize;
      const yPos = y * -100 + height;
      let object: GameObject | undefined;
      switch (cell) {
        case 1:
          object = new FragileBlock(xPos, yPos, xStepSize, 20);
          break;
        case 2:
          object = new Block(xPos, yPos, xStepSize, 20);
          break;
        case 3:
          // bonus item
          object = new Item(
            x * xStepSize,
            y * -100 + height,
            xStepSize,
            imgItemWatermelon,
            30
          );
          break;
        case 4:
          //SpeedBoost item
          object = new SpeedBoost(
            x * xStepSize,
            y * -100 + height,
            xStepSize,
            xStepSize,
            50
          );
          break;
```

UML


GameController
- level: Level
- player: Player:
- levelFactory: LevelFactory
- collisionDetection: CollisionDetection
- score: number
- highscore: number
- levelNumber : number
- isStartingNextLevel: boolean
- countDown: number
- isStartGame: boolean
- isGameOver: boolean
- scoreboard: Scoreboard
- startScreen: StartScreen | undefined
- gameOverScreen: GameOverScreen | undefined
+ GameController(showStartScreen: boolean)
+ update()
+ draw()
- saveHighscore()
- isPlayerDead(): boolean
- startNextLevel()
- displayCountDown()
- updateScore(itemscore: number)

GameController
update()

```
this.level.levelObjects.forEach(levelObject => {
  const isblockCollision = this.collisionDetection.playerCollidedWithBlock(
    this.player,
    levelObject
  );
  const isItemCollision = this.collisionDetection.playerCollidedWithItem(
    this.player,
    levelObject
  );

  if (isblockCollision) {
    if (levelObject instanceof Block) {
      const didBounce = this.player.bounceOnBlock(levelObject.pos);
      if (didBounce) jumpSound.play();
    } else if (levelObject instanceof FragileBlock) {
      if (!levelObject.isDestroyed) {
        const didBounce = this.player.bounceOnBlock(levelObject.pos);
        if (didBounce) levelObject.destroy();
      }
    }
  } else if (isItemCollision) {
    if (levelObject instanceof SpeedBoost) {
      levelObject.applySpeedBoost(this.player);
      this.level.pickUpItem(levelObject);
      this.updateScore(levelObject.points);
    } else if (levelObject instanceof Item) {
      this.level.pickUpItem(levelObject);
      this.updateScore(levelObject.points);
    }
  }
```

REFLEKTIONER

—

- För mycket tid på distans
- Mer planering
- Bättre kommunikation kring design
- Tydligare issues
- Mer kommentarer i koden

+

- Github flow 🏆
- Från spelidé till resultatet
- Samarbetet
- Allmän kommunikation
- Adam