Rapport Platformer

Oleksandr & Morten **15/04/2020**

Brugermanual/instruktion

Vores spil er en lille Platformer hvor det simple mål er at nå så hurtigt som muligt til enden af banen. Dette gøres ved at hoppe på platforme ved hjælp af WASD eller piletasterne. Jo mere man holder sig i bevægelse desto højere fart vil man opnå, dvs. at for at kunne klare spillet hurtigst muligt skal man optimere sin rute og lade være med at kollidere og miste sin fart. Dette koncept minder lidt om måden mange spil gennemføres på under et speedrun.

Kravspecifikationer

Krav

Hårde

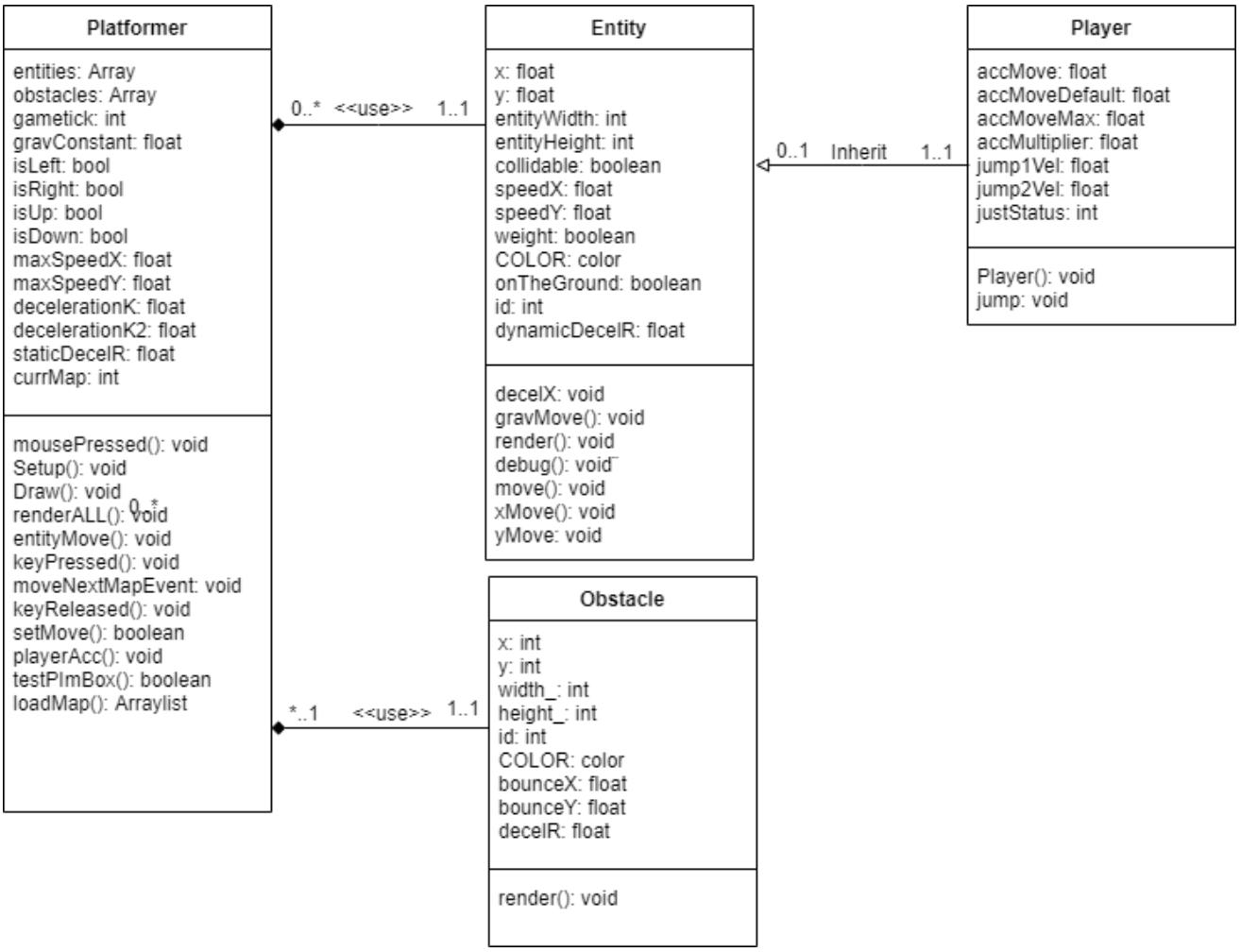
- En entitytype med en player subclass
- Et map bestående af platforme
- player movement
- player collision
- player speeds, acceleration and maxspeed

Bløde

- Grafik - Character

Diverse dokumentationsformer

Klassediagram

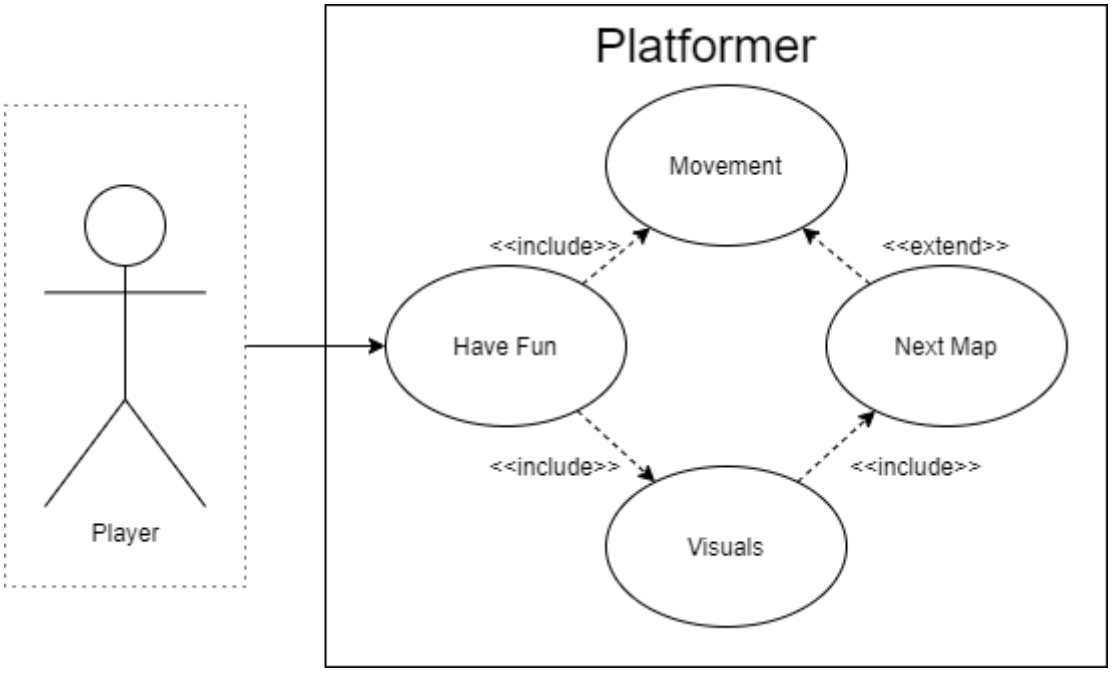


https://drive.google.com/file/d/1cTD7wBTyo160eDofUrQmSIILEGq9ozAq/view?usp=sharing

‘

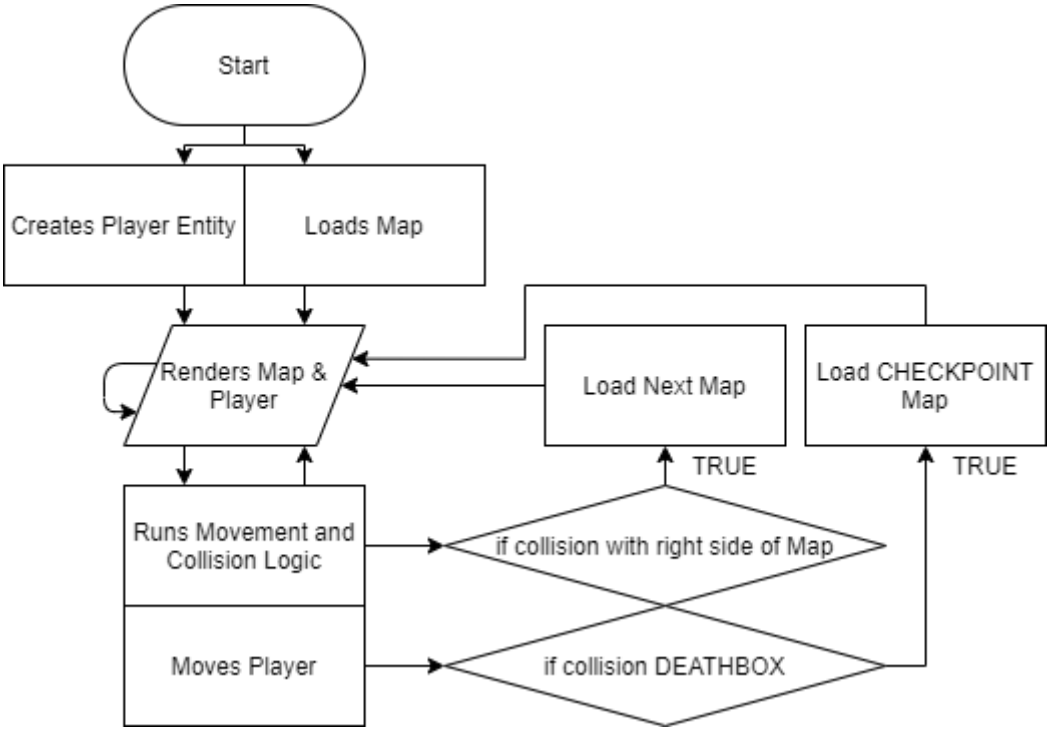
Use case diagram

Et use case diagram giver ikke den store mening med et spil da spil oftest ikke har andre formål end underholdning. Dvs. at et simpelt spil ikke udfører nogen reél handling som vil påvirke “den virkelige verden” på nogen mærkbar måde. Man kunne måske argumentere at det er en use case eks. at flytte spilleren men det er jo ikke et formål i sig selv, da det ikke er grunden til at man ville spille spillet. De fleste spil spillet for underholdning hvilket også er tilfældet med vores. Nogle spil spilles for læring eller sociale forhold men disse er ikke relevante i vores spil. Nedenfor er et forsøg på at lave et use case diagram for spillet.



https://drive.google.com/file/d/1kbcMOx8WB5dm_S2E6UOSzzIRA3ygSZb4/view?usp=sharing

Flowchart



https://drive.google.com/file/d/1B2xtju5V28JahSjPdYXpShbI_wdsOVgn/view?usp=sharing

Beskrevne kodeeksempler

Se i dit foretrukne program for bedre synlighed (er også direkte i koden)

KeyPressed

```
void keyPressed() {
  // This executes if a key is pressed
  // Part of playerAcc
  // While forcing Playertype makes accMove equal to accMove * accMultiplier and
implements MaxMoveMax
  // entities.get(0) gets entity number 0 which is the player
  //.accMove selects the variable accMove which is the accelerated movement (sum of
pos, speed, etc.)
  // This is added to accMultiplier which is the rate at which we go faster (kind
of acceleration)
  // If these are under the maximum speed allowed (accMoveMax)
  // Then the movement is set to their value, if not them the movement is set to
the maximum value
  // setMove executes the jump function if isUp is true, (if arrowUp or W is
pressed)

  if (abs((((Player) entities.get(0)).accMove + ((Player)
entities.get(0)).accMultiplier)) > ((Player) entities.get(0)).accMoveMax) {
    ((Player) entities.get(0)).accMove = ((Player) entities.get(0)).accMoveMax;
  } else {
    ((Player) entities.get(0)).accMove = abs(((Player) entities.get(0)).accMove +
((Player) entities.get(0)).accMultiplier);
  }
  setMove(keyCode, true);
  if (isUp) {
    ((Player)entities.get(0)).jump();
  }
}
```

Interpreter

Inter

```
ArrayList loadMap(String path) {
  String[] lines = loadStrings(path); //læser filen og laver det til en String[]
  String[] strLines = {};
  ArrayList<Obstacle> obstacles = new ArrayList<Obstacle>();//laver en arraylist
til vores obstacles
  Boolean comment; //laver en bool “comment” som specificere om det er en
kommentar.

  //interpreter
  for (String str : lines) { //for hver linje i lines str[]
       comment = false;
       String outputStr = "";  //default output er none
       for (int i = 0; i < str.length(); i++) {
       char curr = str.charAt(i); //den nuværende char
       if (curr == '\"') {
       comment = !comment; //Hvis char == “ så bliver det til comment eller
!comment
       continue;
       }
       if (comment) {
       continue; //hvis det er en kommentar skal man bare kade outputstr = “”
       }
       if (isValidChar(curr)) { //isValidChar er en funktion der checker om det er
a-z eller A-Z eller 0-9 alle ikke valide chars skal frasorteres

       outputStr += curr;
       }
       }
       strLines = append(strLines, outputStr); //strlines er en string[] som
  }
  println("done interpeting");

  //str => int
  for (String strLine : strLines) {
       String[] numberStr = split(strLine, ',');

       if (numberStr.length >= 7) { //x1, y, x2, y2, r, g, b
       obstacles.add(new Obstacle(int(numberStr[0]), int(numberStr[1]),
int(numberStr[2]), int(numberStr[3]), int(numberStr[4]), int(numberStr[5]),
int(numberStr[6])));
       println("ld map RGBmode", numberStr.length, numberStr[4], numberStr[5],
numberStr[6]);
       } else if (numberStr.length == 5) {//x1, y1, x2, y2, #rgb4 (hex)
       obstacles.add(new Obstacle(int(numberStr[0]), int(numberStr[1]),
int(numberStr[2]), int(numberStr[3]), unhex(numberStr[4])));
       println("ld map hexmode", numberStr.length );
       } else if (numberStr.length >= 3) {//x1, y1, x2, y2
       obstacles.add(new Obstacle(int(numberStr[0]), int(numberStr[1]),
int(numberStr[2]), int(numberStr[3])));
       println("ld map posmode", numberStr.length, 255, 255, 255);
       }
  }
```

```
  println("done converting str[] => Obstacle[]");


  String[] slines = loadStrings(path);
  String text = join(slines, "\n");
  String[] NextMapRX =     match(text, "^@\\s*NEXTMAP\\s*=\\s*(.{0,4}\\x2Etxt)$");
  String[] PrevMapRX =     match(text, "^@\\s*PREVMAP\\s*=\\s*(.{0,4}\\x2Etxt)$");
  String[] CheckPointRX =  match(text,
"^@\\s*CHECKPOINT\\s*=\\s*(.{0,4}\\x2Etxt)$");
  //String[] PlayerModelRX = match(text,
"^@\\s*PLAYERMODEL\\s*=\\s*(.{0,18}png)$");
  //String[] BGimageRX =   match(text, "^@\\s*BGIMAGE\\s*=\\s*(.{0,18}\\x2Epng)$");
  //String[] BounceBox =   match(text,
"^@\\s*BOUNCBOX\\s*=\\s*(\\d{0,5}),\\s*(\\d{0,5}),\\s*(\\d{0,5}),\\s*(\\d{0,5})\\s{
0,2}$");
  //String[] DeathBox =    match(text,
"^@\\s*DEATHBOX\\s*=\\s*(\\d{0,5}),\\s*(\\d{0,5}),\\s*(\\d{0,5}),\\s*(\\d{0,5})\\s{
0,2}$");
  nextMap = NextMapRX[1];
  previousMap = PrevMapRX[1];
  checkPoint = CheckPointRX[1];
  println("mapVarsLoaded");
  return obstacles;
}
```

Andet

Art Concept