SNIZARD VS SNIZARD

A PBNJ Game Demo
CS307 04.30.20

Game Concept

-Play as a Snizard (Snake Wizard) or Snizard (Snail Wizard)

-Get from one objective to the other

Design Concept

- A Scrolling Platformer
- Load in Incomplete Level
- In a Builder Stage, allow player to complete level from a bank of objects
- Play User-created Level
- Let Users choose a side

Let Users Customize

Functionality

Let’s play a basic game (Level 1).

Basic Data Files

Two Packages

- User Package
  - Using Firebase
  - Read a json file of Users
  - Create a User class for Pages to manage
  - If user updates, write that back to json file
- LevelData Package
  - Read a json file of Bank objects
  - Read a json file of Level objects
  - Assemble a list of gameobjects to make

User id (username) is the key

Assumption: Month, day, year input (sorry British users)

Score is saved as an int

Password

Paths: a future functionality

Avatar: Which avatar to play as

Type: Which side

Level high scores

users.json | MenuButtons.properties | GraphicsEngine.java | Tester.java | LevelD

```
}
},
"nl163":{
  "birthday":{
    "month":12,
    "year":1999,
    "day":15
  },
  "score":4000,
  "password":"nicole",
  "paths":{
    "1":[1.0,1.0,1.0,1.0,1.0,1.0,11.0,1.0,11.0,1.0]
  },
  "id":"nl163",
  "avatar":"images\/avatars\/gardensnake.png",
  "type":1,
  "levels":{
    "1":55,
    "2":12,
    "3":0
  }
},
```

A User Json file

Key is an int (Level)

Width and height is size of bank in pixels

Money available: how much player can buy

GameObject type

Image Path

Cost

Height/Width according to Grid

Quantity available

A Bank Json file

LevelData.java | banks.json | Main.java | BANK_PURCHASE_BUG.md | R

```
{
  "0": {
    "width" : 200,
    "height" : 280,
    "moneyAvailable" : 1000,
    "items" : {
      "engine.gameobject.platform.StationaryPlatform" : {
        "ctor": [
          ["java.lang.String", "images/objects/grasstile.png"],
          ["java.lang.Double", "1"],
          ["java.lang.Double", "1"],
          ["java.lang.Double", "-1"],
          ["java.lang.Double", "-1"]
        ],
        "cost": 20,
        "height": 1,
        "width": 1,
        "quantity": 2
      },
      "engine.gameobject.platform.VerticalSlidingPlatform" : {
        "ctor": [
          ["java.lang.String", "images/objects/broomstick.png"],
          ["java.lang.Double", "1"],
          ["java.lang.Double", "1"],
          ["java.lang.Double", "-1"],
          ["java.lang.Double", "-1"],
```

How the Bank looks

Key is an int (Level)

Game Object Type

Image Path

Height (relative to Grid)

Width (relative to Grid)

X position

Y position

banks.json | levels.json | Main.java | BANK_PURCHASE_

```
{
  "0":{
    "engine.gameobject.platform.StationaryPlatform":[
      [
        ["java.lang.String","images\/objects\/grasstile.png"],
        ["java.lang.Double","1.0"],
        ["java.lang.Double","1.0"],
        ["java.lang.Double","2.0"],
        ["java.lang.Double","3.0"]
      ],
      [
        ["java.lang.String","images\/objects\/grasstile.png"],
        ["java.lang.Double","1.0"],
        ["java.lang.Double","1.0"],
        ["java.lang.Double","4.0"],
        ["java.lang.Double","25.0"]
      ],
      [
        ["java.lang.String","images\/objects\/grasstile.png"],
        ["java.lang.Double","1.0"],
        ["java.lang.Double","1.0"],
```

A Level Json file

How the Level looks

That’s how the Data Package works!

Tests

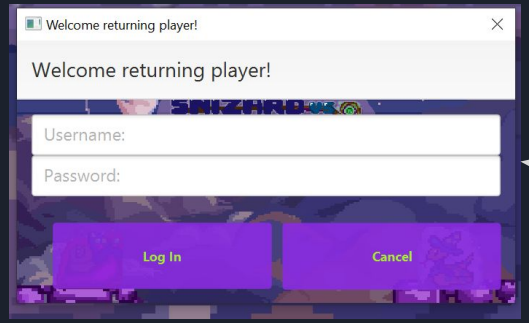
DESIGN

Flexibility

- Reflection for data inputs
  - Effect: No hard-coded Levels
- CSS styling
  - Effect: No hard-coded Styling
- Model-View-Controller
  - Effect: Easily Modifiable Features

Use Case 1: Build a New User

Option 1: Player continues pre-existing game, initiating a pop-up login dialogue.



Option 2: User presses Exit Button, exiting the program.

Option 3: User starts a New Game, starting a new Scene.

Loading the Game, Logging in

The user's name

The user's score

Avatar the user selected

Level Directory → let's try a new user in master

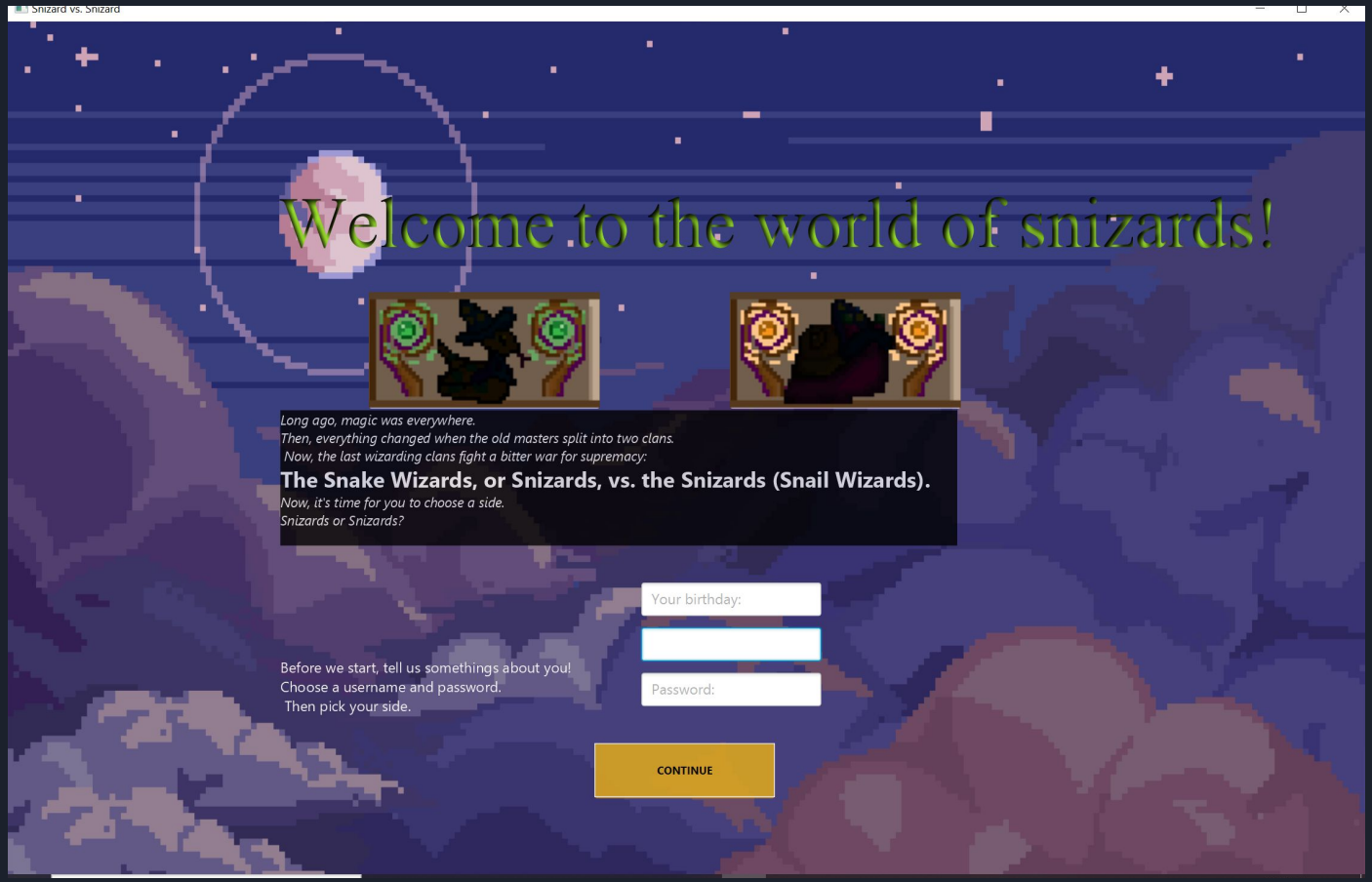
Getting the Necessary Information for a User

1) A Birthday (Numeral Dates or Strings)

2) A Username (must not already exist)

3) Any password of at least one string length

4) A user must choose a side (Snizard [Snake Button] or Snizard [Snail button]



Making a New Game

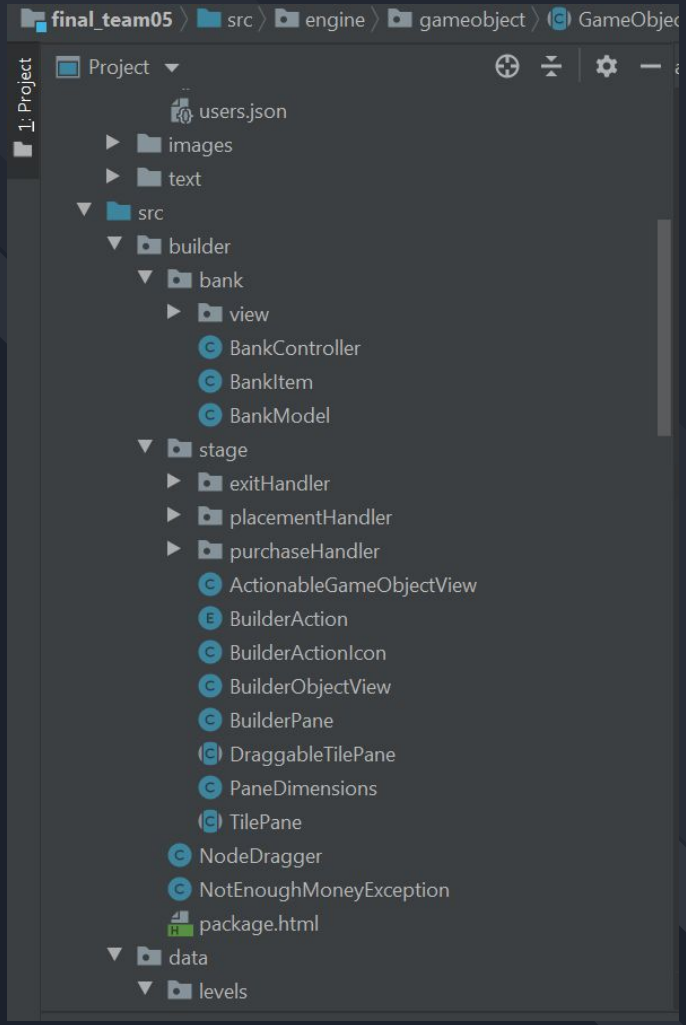
How did our design change?

Design goals

No GameObjects are hardcoded

- Our functionality hardcoded (Horizontal Sliding Platforms move horizontally) BUT NOT THE IMAGEVIEW associated. Any image loaded into a GameObject can function if assigned as a Horizontal Sliding Platform, etc.
- To do this, we used JSON files that recorded GAME BEHAVIOR and IMAGEPATH
  - This allows us to maybe one day add DLC and custom skins
  - Change the art at any time

Use Case 2 : Builder Stage



Sad Paths:

- User doesn’t remember password
- User puts blocks over each other

TEAM

Timeline

- Build Page class structure
- Json data files
- Builder Stage
- Play Level and Game Object interactions

Things we learned

- One thing we Learned about Project Management
- One thing we could still improve
- One thing we learned about Positive Team Culture
- One thing we learned about Communicating


LIGHT MODE
DARK MODE