Team Name: Team FooBar

The team members of our group and tentative division between BackEnd and FrontEnd for game engine and game editor are as below:

Backend:

- Wenxin Shi
- Fabio Berger
- Yuhua Mai
- Harris Osserman
- Jiaran Hao

Frontend:

- Susan Zhang
- Rebecca Lai
- Lalita Maraj
- Alex Zhu

We have discussed and compared the following 3 genres and decided to make a Tower Defense game. The difficulty of making a Tower Defense game is between making a scrolling game and a real time strategy game.

Tower Defense:

Requires:

- Towers
- Paths
- Treasure
- Enemies
- Spawn rates
- Cheat codes

Metaphor:

- Play until you lose all treasure
- Different waves of enemies

Design Goals:

- Reading from a user-configured text to initialize the game, using reflection to create objects dynamically
- Having flexible/configurable user interface
- Ability to dynamically add/remove code

Examples:

http://www.coolmath-games.com/0-bloons-tower-defense-1/
Goal: Survive as many waves of attack as you can.
Rule:
1 Lose when you lose all life
2 Survive fixed rounds
Strategy: Use supply of towers to block enemies from your castle with the constraint of limited money. The more enemies you block, the more money you gain.

Path: User Define, Check valid path, Multiple Layers(Tunnel, Bridges), Unchangable within game
    Level: Formula to design different levels
        Tower: Power, Range, Property(Fire, Ice, Poison...)
        Enemy: Life, Speed, Property(Stone, …)
        Statistic: Life, Money, …

Game Data

Game Authoring group and Game Engine group will develop their part according to the Game Data.

Represent by 2D array map, each object will fill in one spot. View could be smaller than the map.

- = variable + = method

Library
- - Tower
- + getAvaibleTower

Tower
Attributes user can change
- - Image - how tower is graphically represented (1)
- - Damage (1)
- - Attack speed (1)
- - Location (1)
- - Range (1)
- - Cost (1)
- - Recycle Price (2)
- - Life (2) - life left after enemy damages

Enemy
- - Gold Bonus - gold you gain from killing that enemy(1)

  - Life - how resilient the enemy is (2)
  - Moving Speed - how fast they move (2)
  - Damage (2) - damage that can be inflicted on a tower
  - Time of spawn (1)

Map
  - Background img (1)
  - Dimension (of the playing grid) (2)
  - Path (with start and end) (1)

Game info
  - Name (1)
  - Splash page image (1)
  - Number of levels (1)
  - Level difficulty increase (1)
  - Choose enemy type for each level (1)

User:
  - Gold
  - Number of lives (1)

how does game get harder?

Group Division

Backend:
-

JGObject
- behavior in Model

JSON
API

- **Genre**
    -
- **Describe your game genre and what qualities make it unique that your design will need to support**
- **Design Goals**
- **Describe the goals of your project's design (i.e., what you want to make flexible and what assumptions you may need to make), the structure of how games are represented, and how these work together to support a variety of games in your genre.**
- **Primary Classes and Methods for each module**
- **Describe the program's core architecture (focus on behavior not state), including pictures of a UML diagrams and "screen shots" of your intended user interface. Each sub-project should have its own API for others in the overall team.**
- **Example code**
- **Rather than providing specific Java code, you should describe two or three example games from your genre in detail that differ significantly and the data files that represent them and could be saved and loaded by your project. You should use these examples to help make concrete the abstractions you have identified in your design.**
- **Explain some alternatives to your design, and why you choose the one you did**
- **List of each team member's role in the project and a breakdown of what each person is expected to work on.**

Game Engine API

//Tower
createTower(Position, TowerType)
destroyTower(Position)

Enemy

- Gold Bonus - gold you gain from killing that enemy(1)
- Life - how resilient the enemy is (2)
- Moving Speed - how fast they move (2)
- Damage (2) - damage that can be inflicted on a tower
- Time of spawn (1)

Map

- Background img (1)
- Dimension (of the playing grid) (2)
- Path (with start and end) (1)

Game info

- Name (1)
- Splash page image (1)
- Number of levels (1)
- Level difficulty increase (1)
- Choose enemy type for each level (1)

Rule

-Decide win condition or lose condition
-Generate enemies based on the configuration of the authors.
-addWave()
- may decide if the user can buy certain towers.

Wave

- Wave is the class that store the information of different Waves of enemies in the tower defense game.
- one can set spawn rate, type of enemy, interval between two Waves, number of enemy in Wave.

Audio

-Register the audio based on the configuration of the authors
so that all the classes can simply call

User:

- Gold
- Number of lives (1)

JSON
Example:

```
{
        “name”: “Tower Destruction II”,
        “spashImage” :  “http://oogasalad.com/splash.png”,
        “gold” : 200,
        “numberOfLives” : 100,
         “levels”: 5,
        “difficultyScale”: 1.5,

        “map”:
       {
          “BGImage”:“http://oogasalad.com/towerImg1.png”,
           “PathImage”:  “http://oogasalad.com/path.png”,
          “Path”:[
                   {  “x”:1, “y”:5},
                   { “x”: 4, “y”: 8 }
                ]
        },


        “towerType”: [
                {
                 “ID”: “1”
                 “image”: “/resources/images/towerImg1.png”,
                 “damage”: 5,
                 “attackSpeed”: 3,
                “range”: 3,
                “cost”: 10,
                “recyclePrice”:  5,
                ”life”: 2
                }
        ],

        “enemyType”:[
                {
                    “gold”: 5,
                    “image”: “/resources/images/enemyImg1.png”,
                    “life”: 10,
                    “speed”: 20,
                    “damage”: 40,
                    “time”: 3
                },
                {
                    “gold”: 25,
                    “image”: “/resources/images/enemyImg2.png”,
                    “life”: 20,
                    “speed”: 30,
                    “damage”: 60,
                    “time”: 5
                }
        ],


        “levelData”: [
                {
                    “level”:1
                    “numberWaves”: 2,
                    “enemiesOfWave” : [
                                    {“wave”:1,
                                     enemies:[
                                            {“id”:1,“quantity”: 45},
                                            {“id”:2,“quantity”:50}
                                         ]
                    ]

                }

        ]


}
```

View Primary Classes and Methods

```
class Game extends JGEngine{








}

class Controller{
      //Only acts on the View and Model not subcomponents of the View
      //and Model
      Model model;
      View view;

      public void newGame(File jsonFile);

      public void setMoney(int money);
      public int getMoney();

      public int getLife();

      public void setTower(Type type, Position pos);
      public List<Tower> getTowers();

      public List<Enemy> getEnemies();
      public List<Bullet> getBullets();

      public List<Path> getPath();

      public Image getPathImage();
      public Image getBackgroundImage();
}
```

Game Authoring Primary Classes and Methods:

```
class GameData
class GameAuthoringGUI
class MapDesignTab
class TowerDesignTab
class EnemyDesignTab
class LevelDesignTab
```