CONCEPTIONS

Personal Modifications:

The biggest change of the architecture that has been made is about the abstract class "Level,". In our project, the Level class lost its abstract property substantially. We have noticed that every Level should have a Bike, StartFlag, FinishLine and a Terrain as its properties to a certain extent. That's why we have chosen to create these properties in Level class rather than implementing them in every single subclass. The level class has two overloaded methods called **createAllActors()**. The method without parameter is an abstract method and only provides a template to the subclasses. The other method functions like a super constructor and initializes all the properties.

-We have added constaint builders to ActorGame to avoid writing an accessor for world(getWorld()) as recommended.

-Another added notion is ContactSituation , this notion connected to the listener is used for seperating the cases for which bike hits.

If contactSituation() == 1 bike hits a ghost

If contactSituation() == 0 bike hits a concrete object

If contactSituation() == -1 bike didn't hit anything

This division is useful for some cases where we avoided false game endings in cases of collision between triggers etc.

Extensions:

Levels: Game is divided into levels as mentioned before.

Trigger: A trigger super class is written as a template for all ghost entities in the game.

FinishLine: Class derived from Trigger. If finishline has been triggered, the level is succeded.

Coins and coinScore: Coin is a derived class of Trigger. Coins are collectable objects and coin score shows number of coins collected.

Pendulum: Rope class(Level3, Level5, Level6)

Revolute: Revolute class(Level2, Level6)

Trampoline: Trampoline class (Level3)

Gravity Well: GravityField class (Level6)

Slippery Terrain: Slippery class (Level5) Instead of making the terrain slippery by decreasing friction we implemented a floor which hits back the bike, to challenge the player. (see README)

Pics : When taken, gives some powers or creates objects to pass or create an obstacle.

Animation of Cyclist: Pedaling(Bike class doPedaling()) & Arms up (Bike class setVictoryPosition()).

Extras

Fire : A GameEntity which should be avoided by the biker

Missile: A GameEntity should be avoided by the biker

SpeedBoost : Attached to a trigger, when a star(collectable) is taken gives the enough impulse to pass obstacle.

(For the game logic used in every extension please refer to README , whilst explaining the walkthrough of the game, we explained what their purposes are and how they integrate in the game)

Hope you enjoy it !

Kagan Bakirci – Cagin Tanir