- concept for this text-oriented View is not clear here

- characters cannot be requested to get displayed with the current approsach (characters are not specializations of game elements)

- when the model needs to have things displayed, are these requests done on a per game element basis? screen? scene? act? game?

- not clear how this is stubbing user inputs (e.g., mouse click), displaying outputs, removing outputs...

- view needs to know about starting, playing, ending acts, scenes, screens, challenges? or does it? design concept for the View not clear, which has a significant impact on the interfaces, Model, and Controller.

- UI needs to definitely know about what to display, what to remove from the display, what can be deleted - e.g. if we are ending a screen, all of the entities created for the screen are no longer needed...)

/\*\*

\* Dummy View class implementation - Will be replaced by actual view classes - this should be a text version of the View - otherwise, cannot see if the GamePlayEngine model is behaving correctly.

\*/

package gamePlayEngine.view;

import gamePlayEngine.controller.GamePlayEngineController;

import gamePlayEngine.controller.MessageType;

import gamePlayEngine.model.gameElement.graphic.Backdrop;

import gamePlayEngine.model.gameElement.graphic.Prop;

import gamePlayEngine.model.gameElement.player.reward.Reward;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Observable;

import java.util.Observer;

public class GameView implements Observer {

private GamePlayEngineController controller;

private String displayInfo;

public GamePlayEngineController getController() {

return controller;

}

public void setController(GamePlayEngineController controller) {

this.controller = controller;

}

suggest using a different naming convention for the "props" in the View - otherwise there will be props in the Model, props in the View - confusing.

- this View is using text o/p to stub the functionality - could call them textXxx to distinguish.

@Override

public void update(Observable arg0, Object arg1) {

if (Prop.class.isInstance(arg0)) {

DrawProp((Prop) arg0);

} else if (Backdrop.class.isInstance(arg0)) {

DrawProp((Backdrop) arg0);

} else if (Reward.class.isInstance(arg0)) {

displayInfo = Integer.toString(((Reward) arg0).getPoints());

}

}

comment???

private void DrawProp(Backdrop backdrop) {

System.out.println("Creating view for Backdrop");

System.out.println("Loading backdrop image from path "

+ backdrop.getPath());

}

comment???

private void DrawProp(Prop prop) {

if (prop.getTypeName().equals("java.awt.Button")) {

DrawButton(prop);

} else if (prop.getTypeName().equals("java.awt.Label")) {

DrawLabel(prop);

}

}

comment???

private void DrawLabel(Prop prop) {

System.out.println("Creating view for Prop (" + prop.getTypeName()

+ ") with text " + prop.getText() + " " + displayInfo);

}

comment???

private void DrawButton(Prop prop) {

System.out.println("Creating view for Prop (" + prop.getTypeName()

+ ") with text " + prop.getText());

System.out.println("View: Please enter user input: ");

try {

String line;

BufferedReader is = new BufferedReader(new InputStreamReader(

System.in));

line = is.readLine();

controller.play(MessageType.External,prop.getBehavior());

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}