**Added an if statement to check if we are in the Inventory state in the drawEnergies method. Also added the inv local variable to the Inventory if statement in the drawEnergies method in the GameState class.**

**package** sonar.gamestates;

**import** java.awt.Graphics;

**import** java.awt.image.BufferedImage;

**import** java.io.IOException;

**import** javax.imageio.ImageIO;

**import** sonar.gamestates.states.Inventory;

**import** sonar.gamestates.states.levels.stages.entities.SpriteManager;

**import** sonar.gamestates.states.levels.stages.entities.animations.tiles.Tile;

**import** sonar.gamestates.states.levels.stages.entities.animations.tiles.TileManager;

**public** **abstract** **class** GameState

{

//The base class Template for all the gamestates in the game.

**private** StateBuilder buildState;

**private** GSM gsm;

**private** Keyboard key;

**private** SpriteManager smanage;

**private** TileManager tmanage;

**private** **int**[] tiles;

**private** **int** width, height;

**private** String identity;

**protected** GameState(StateBuilder buildState, String path, String identity, GSM gsm)

{

**this**.buildState = buildState;

**this**.gsm = gsm;

**this**.identity = identity;

**if**(buildState.stateType().equals("Single"))

{

key = **new** Keyboard(gsm);

smanage = **new** SpriteManager(identity);

tmanage = **new** TileManager(smanage);

}

**if**(!identity.equals("Starter")) createGameState(path);

}

**private** **void** createGameState(String path)

{

**try**

{

BufferedImage image = ImageIO.*read*(GameState.**class**.getResource(path));

width = image.getWidth();

height = image.getHeight();

tiles = **new** **int**[width \* height];

image.getRGB(0, 0, width, height, tiles, 0, width);

}

**catch** (IOException e){e.printStackTrace();}

}

**protected** **void** update()

{

}

**protected** **void** render(**int** xScroll, **int** yScroll, Screen screen, Graphics g)

{

screen.setOffset(xScroll, yScroll);

drawTiles(xScroll, yScroll, screen);

drawWeapons(screen);

drawEnergies(screen);

}

**private** **void** drawEnergies(Screen screen)

{

**if**(identity.equals("Inventory"))

{

Inventory inv = (Inventory) gsm.getCurrentState();

}

}

**private** **void** drawWeapons(Screen screen)

{

**if**(identity.equals("Inventory"))

{

Inventory inv = (Inventory) gsm.getCurrentState();

inv.renderWeapons(screen);

}

}

**public** **void** drawTiles(**int** xScroll, **int** yScroll, Screen screen)

{

**if**(tmanage != **null**)

{

**int** x0 = xScroll / tmanage.voidTile.getWidth(); //divided by 16

**int** x1 = (xScroll + screen.getWidth() + tmanage.voidTile.getWidth()) / tmanage.voidTile.getWidth();

**int** y0 = yScroll / tmanage.voidTile.getHeight();

**int** y1 = (yScroll + screen.getHeight() + tmanage.voidTile.getHeight()) / tmanage.voidTile.getHeight();

drawGameState(x0, x1, y0, y1, screen);

}

}

**private** **void** drawGameState(**int** x0, **int** x1, **int** y0, **int** y1, Screen screen)

{

**for**(**int** y = y0; y < y1; y++)

{

**for**(**int** x = x0; x < x1; x++)

{

getTile(x, y).render(x, y, screen);

}

}

}

Tile getTile(**int** x, **int** y)

{

Tile tile = tmanage.voidTile;

**if**(x < 0 || y < 0 || x >= width || y >= height) **return** tile;

**if**(identity.equals("Menu"));

**if**(identity.equals("Password")) tile = invpassCommons(identity, x, y, tile);

**if**(identity.equals("Inventory")) tile = invpassCommons(identity, x, y, tile);

**if**(identity.equals("Starter"))

{

**if**(tileColour(x, y) == TileManager.***grassColour***) tile = tmanage.grass;

}

**return** tile;

}

**private** Tile invpassCommons(String identity, **int** x, **int** y, Tile tile)

{

**if**(tileColour(x, y) == TileManager.***cornerUpLeftColour***) tile = tmanage.cornerUpLeft;

**if**(tileColour(x, y) == TileManager.***cornerUpRightColour***) tile = tmanage.cornerUpRight;

**if**(tileColour(x, y) == TileManager.***cornerDownLeftColour***) tile = tmanage.cornerDownLeft;

**if**(tileColour(x, y) == TileManager.***cornerDownRightColour***) tile = tmanage.cornerDownRight;

**if**(tileColour(x, y) == TileManager.***lineUpColour***) tile = tmanage.lineUp;

**if**(tileColour(x, y) == TileManager.***lineDownColour***) tile = tmanage.lineDown;

**if**(tileColour(x, y) == TileManager.***lineLeftColour***) tile = tmanage.lineLeft;

**if**(tileColour(x, y) == TileManager.***lineRightColour***) tile = tmanage.lineRight;

**if**(tileColour(x, y) == TileManager.***squareColour***) tile = tmanage.square;

**return** tile;

}

**private** **int** tileColour(**int** x, **int** y){**return** tiles[x + y \* width];}

StateBuilder getBuildState(){**return** buildState;}

**protected** GSM getGsm(){**return** gsm;}

**protected** Keyboard getKey(){**return** key;}

**protected** **void** resetKeyboard(){key = **null**;}

**protected** **void** initKey(){key = **new** Keyboard(gsm);}

**public** SpriteManager getSmanage(){**return** smanage;}

**public** TileManager getTmanage(){**return** tmanage;}

**protected** **void** resetSmanage(){smanage = **null**;}

**protected** **void** setSmanage(SpriteManager manage){smanage = manage;}

**protected** **void** resetTmanage(){tmanage = **null**;}

**protected** **void** setTmanage(TileManager manage){tmanage = manage;}

**protected** String getIdentity(){**return** identity;}

**public** **void** setTiles(**int**[] tiles){**this**.tiles = tiles;}

**public** **void** setWidth(**int** value){width = value;}

**public** **void** setHeight(**int** value){height = value;}

}