**Initialized cornerDownRight and lineUp variables to the appropriate sprites.**

**package** sonar.gamestates.states.levels.stages.entities;

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

**import** java.io.IOException;

**import** javax.imageio.ImageIO;

**public** **class** SpriteManager

{

//Manages the sprites for all objects

**public** Sprite voidSprite, grass;

**public** Sprite cornerUpLeft, cornerUpRight, cornerDownLeft, cornerDownRight;

**public** Sprite lineUp, lineDown, lineLeft, lineRight;

**private** String identity;

**public** SpriteManager(String identity)

{

**this**.identity = identity;

**if**(identity.equals("Menu")) buildMenu(**new** SpriteSize(16, 16));

**if**(identity.equals("Password")) buildPassword(**new** SpriteSize(16, 16));

**if**(identity.equals("Inventory")) buildInventory(**new** SpriteSize(16, 16));

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

}

**private** **void** buildMenu(SpriteSize size)

{

voidSprite = **new** Sprite(**new** HUDSpriteBuilder(size, 0xffff0000));

}

**private** **void** buildPassword(SpriteSize size)

{

voidSprite = **new** Sprite(**new** HUDSpriteBuilder(size, 0xff00ff00));

}

**private** **void** buildInventory(SpriteSize size)

{

voidSprite = **new** Sprite(**new** HUDSpriteBuilder(size, 0xff00ffff));

SpriteSheet inventory = **new** SpriteSheet("/textures/tiles/InventorySheet.png");

cornerUpLeft = **new** Sprite(**new** ColourSpriteBuilder(size, **new** SpriteLocation(0, 0, inventory)));

cornerUpRight = **new** Sprite(**new** ColourSpriteBuilder(size, **new** SpriteLocation(1, 0, inventory)));

cornerDownLeft = **new** Sprite(**new** ColourSpriteBuilder(size, **new** SpriteLocation(0, 1, inventory)));

cornerDownRight = **new** Sprite(**new** ColourSpriteBuilder(size, **new** SpriteLocation(1, 1, inventory)));

lineUp = **new** Sprite(**new** ColourSpriteBuilder(size, **new** SpriteLocation(2, 0, inventory)));

}

**private** **void** buildStarter(){buildTileSprites();}

**private** **void** buildTileSprites()

{

buildStaticTiles();

buildDynamicTiles();

}

**private** **void** buildStaticTiles()

{

voidSprite = **new** Sprite(**new** HUDSpriteBuilder(**new** SpriteSize(16, 16), 0xff0000ff));

grass = **new** Sprite(**new** ColourSpriteBuilder(**new** SpriteSize(16, 16), **new** SpriteLocation(0, 0, **new** SpriteSheet("/textures/tiles/SpriteSheet.png"))));

}

**private** **void** buildDynamicTiles()

{

}

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

}

**abstract** **class** SpriteBuilder

{

**private** SpriteSize size;

**private** **int**[] pixels;

SpriteBuilder(SpriteSize size)

{

**this**.size = size;

pixels = **new** **int**[size.getWidth() \* size.getHeight()];

}

**int** getWidth(){**return** size.getWidth();}

**int** getHeight(){**return** size.getHeight();}

**int**[] getPixels(){**return** pixels;}

**abstract** **int** hiddenColour();

**abstract** String spriteType();

}

**class** HUDSpriteBuilder **extends** SpriteBuilder

{

HUDSpriteBuilder(SpriteSize size, **int** colour)

{

**super**(size);

loadHUD(size, colour);

}

**private** **void** loadHUD(SpriteSize size, **int** colour)

{

**for**(**int** y = 0; y < size.getHeight(); y++)

{

**for**(**int** x = 0; x < size.getWidth(); x++)

{

getPixels()[x + y \* size.getWidth()] = colour;

}

}

}

**int** hiddenColour(){**return** 0xffff00ff;}

String spriteType(){**return** "HUD";}

}

**class** ColourSpriteBuilder **extends** SpriteBuilder

{

ColourSpriteBuilder(SpriteSize size, SpriteLocation location)

{

**super**(size);

loadColour(size, location);

}

**private** **void** loadColour(SpriteSize size, SpriteLocation location)

{

**int** xw = location.getX() \* size.getWidth();

**int** yh = location.getY() \* size.getHeight();

**for**(**int** y = 0; y < size.getHeight(); y++)

{

**for**(**int** x = 0; x < size.getWidth(); x++)

{

getPixels()[x + y \* size.getWidth()] = location.getSheet().getPixels()[(x + xw) + (y + yh) \* location.getSheet().getSize()];

}

}

}

**int** hiddenColour(){**return** 0xffff00ff;}

String spriteType(){**return** "Colour";}

}

**class** GreySpriteBuilder **extends** SpriteBuilder

{

**private** **int**[] colours;

GreySpriteBuilder(SpriteSize size, SpriteLocation location, **int**... colours)

{

**super**(size);

**this**.colours = colours;

loadGrey(size, location);

}

**private** **void** loadGrey(SpriteSize size, SpriteLocation location)

{

**int** xw = location.getX() \* size.getWidth();

**int** yh = location.getY() \* size.getHeight();

**for**(**int** y = 0; y < size.getHeight(); y++)

{

**for**(**int** x = 0; x < size.getWidth(); x++)

{

getPixels()[x + y \* size.getWidth()] = location.getSheet().getPixels()[(x + xw) + (y + yh) \* location.getSheet().getSize()];

}

}

}

**int**[] getColours(){**return** colours;}

**int** colourSize(){**return** colours.length;}

**int** hiddenColour(){**return** 0xff000000;}

String spriteType(){**return** "Grey";}

}

**class** SpriteLocation

{

**private** **int** x, y;

**private** SpriteSheet sheet;

SpriteLocation(**int** x, **int** y, SpriteSheet sheet)

{

**this**.x = x;

**this**.y = y;

**this**.sheet = sheet;

}

**int** getX(){**return** x;}

**int** getY(){**return** y;}

SpriteSheet getSheet(){**return** sheet;}

}

**class** SpriteSheet

{

**private** **int**[] pixels;

**private** **int** size;

SpriteSheet(String path)

{

loadSheet(path);

}

**private** **void** loadSheet(String path)

{

BufferedImage image;

**try**

{

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

**int** width = image.getWidth();

**int** height = image.getHeight();

size = width;

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

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

}

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

}

**int**[] getPixels(){**return** pixels;}

**int** getSize(){**return** size;}

}

**class** SpriteSize

{

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

SpriteSize(**int** width, **int** height)

{

**this**.width = width;

**this**.height = height;

}

**int** getWidth(){**return** width;}

**int** getHeight(){**return** height;}

}