**Removed the old broken getGSM method since it was no longer needed and also changed the new Keyboard parameter to the getGsm method call in the createGameState method located in the GameState class.**

**package** sonar;

**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** **static** StateBuilder *buildState*;

**private** **static** GSM *gsm*;

**private** **static** Keyboard *key*; //Can be made static

**private** **static** SpriteManager *smanage*; //Can be made static

**private** **static** TileManager *tmanage*; //Can be made static

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

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

**protected** **final** **static** **void** createGameState(**final** StateBuilder cBuild)

{

*buildState* = cBuild;

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

{

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

*smanage* = **new** SpriteManager(*buildState*.getIdentity());

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

}

**if**(!*buildState*.getIdentity().equals("Starter"))

{

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

{

DualStateBuilder dual = (DualStateBuilder) *buildState*;

*loadPath*(dual.getPath());

}

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

{

SingleStateBuilder single = (SingleStateBuilder) *buildState*;

*loadPath*(single.getPath()); //Added loadPath

}

}

}

**private** **final** **static** **void** loadPath(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**(*buildState*.getIdentity().equals("Inventory"))

{

Inventory inv = (Inventory) GSM.*getCurrentState*();

inv.renderEnergies(screen);

}

}

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

{

**if**(*buildState*.getIdentity().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**(*buildState*.getIdentity().equals("Menu"));

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

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

**if**(*buildState*.getIdentity().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** **final** **static** **int** tileColour(**final** **int** x, **final** **int** y){**return** *tiles*[x + y \* *width*];}

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

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

**final** **static** **void** setGSM(GSM cGSM){*gsm* = cGSM;}

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

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

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

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

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

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

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

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

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

**public** **final** **static** **void** setTiles(**int**[] values){*tiles* = values;}

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

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

**final** **static** **void** setBuildState(StateBuilder state){*buildState* = state;}

}