**Initialized lineLeft and lineRight variables to use the appropriate sprites defined by the manage variable.**

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

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

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

**import** sonar.gamestates.states.levels.stages.entities.animations.AnimationType;

**import** sonar.gamestates.states.levels.stages.entities.animations.DynamicAnimation;

**import** sonar.gamestates.states.levels.stages.entities.animations.StaticAnimation;

**public** **class** TileManager

{

//Creates all the tiles necessary in the game.

**public** Tile voidTile, grass;

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

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

**public** **static** **final** **int** ***grassColour*** = 0xff5eb429; //94 red, 180 green, 41 blue

**public** **static** **final** **int** ***cornerUpLeftColour*** = 0xff282828;

**public** TileManager(SpriteManager manage)

{

voidTile = **new** Tile(**new** VoidTileBuilder(**new** StaticAnimation(manage.voidSprite)));

**if**(manage.getIdentity().equals("Menu")) buildMenu(manage);

**if**(manage.getIdentity().equals("Password")) buildPassword(manage);

**if**(manage.getIdentity().equals("Inventory")) buildInventory(manage);

**if**(manage.getIdentity().equals("Starter")) buildStarter(manage);

}

**private** **void** buildMenu(SpriteManager manage)

{

}

**private** **void** buildPassword(SpriteManager manage)

{

}

**private** **void** buildInventory(SpriteManager manage)

{

cornerUpLeft = **new** Tile(**new** CornerTileBuilder(**new** StaticAnimation(manage.cornerUpLeft)));

cornerUpRight = **new** Tile(**new** CornerTileBuilder(**new** StaticAnimation(manage.cornerUpRight)));

cornerDownLeft = **new** Tile(**new** CornerTileBuilder(**new** StaticAnimation(manage.cornerDownLeft)));

cornerDownRight = **new** Tile(**new** CornerTileBuilder(**new** StaticAnimation(manage.cornerDownRight)));

lineUp = **new** Tile(**new** LineTileBuilder(**new** StaticAnimation(manage.lineUp)));

lineDown = **new** Tile(**new** LineTileBuilder(**new** StaticAnimation(manage.lineDown)));

lineLeft = **new** Tile(**new** LineTileBuilder(**new** StaticAnimation(manage.lineLeft)));

lineRight = **new** Tile(**new** LineTileBuilder(**new** StaticAnimation(manage.lineRight)));

}

**private** **void** buildStarter(SpriteManager manage)

{

grass = **new** Tile(**new** GrassTileBuilder(**new** StaticAnimation(manage.grass)));

}

}

**abstract** **class** TileBuilder

{

**private** AnimationType buildAnimation;

TileBuilder(AnimationType buildAnimation){**this**.buildAnimation = buildAnimation;}

**void** update()

{

**if**(buildAnimation.animType().equals("Dynamic"))

{

DynamicAnimation anim = (DynamicAnimation) buildAnimation;

anim.update();

buildAnimation = anim;

}

}

Sprite getSprite(){**return** buildAnimation.getSprite();}

**int** getWidth(){**return** buildAnimation.getSprite().getWidth();}

**int** getHeight(){**return** buildAnimation.getSprite().getHeight();}

**boolean** solid(){**return** **false**;}

**boolean** light(){**return** **false**;}

**abstract** String attribute();

}

**class** VoidTileBuilder **extends** TileBuilder

{

VoidTileBuilder(AnimationType buildAnimation){**super**(buildAnimation);}

String attribute(){**return** "";}

}

**class** GrassTileBuilder **extends** TileBuilder

{

GrassTileBuilder(AnimationType buildAnimation){**super**(buildAnimation);}

**boolean** solid(){**return** **true**;}

String attribute(){**return** "";}

}

**class** CornerTileBuilder **extends** TileBuilder

{

CornerTileBuilder(AnimationType buildAnimation){**super**(buildAnimation);}

String attribute(){**return** "Corner";}

}

**class** LineTileBuilder **extends** TileBuilder

{

LineTileBuilder(AnimationType buildAnimation){**super**(buildAnimation);}

String attribute(){**return** "Line";}

}