**Added a static TileManager for use with colours only.**

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

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

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

**public** **class** Level

{

**private** LM lm;

**private** LevelBuilder buildLevel;

**protected** Level(LevelBuilder buildLevel, LM lm)

{

**this**.buildLevel = buildLevel;

**this**.lm = lm;

}

**void** update()

{

}

**void** render(**int** xScroll, **int** yScroll, Screen screen)

{

screen.setOffset(xScroll, yScroll);

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

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

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

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

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

}

**private** **void** drawLevel(**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 = lm.getTmanager().voidTile;

**if**(x < 0 || y < 0 || x >= buildLevel.getWidth() || y >= buildLevel.getHeight());

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

**return** tile;

}

**private** **int** tileColour(**int** x, **int** y)

{

**return** buildLevel.getTiles()[x + y \* buildLevel.getWidth()];

}

LM getLm(){**return** lm;}

LevelBuilder getBuildLevel(){**return** buildLevel;}

}