**Added colourSize local variable to the render method and set equal to the sprites colourSize for tiles.**

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

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

**public** **class** Screen

{

**private** **int** width, height, xOffset, yOffset;

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

Screen(**int** width, **int** height)

{

**this**.width = width;

**this**.height = height;

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

}

**void** clear()

{

**for**(**int** i = 0; i < pixels.length; i++){pixels[i] = 0;}

}

**public** **void** renderTile(Tile tile)

{

render(tile, tile.getX(), tile.getY(), tile.getWidth(), tile.getHeight());

}

**private** **void** render(Entity entity, **int** xp, **int** yp, **int** eWidth, **int** eHeight)

{

xp -= xOffset;

yp -= yOffset;

**int** hiddenColour = 0;

**for**(**int** y = 0; y < eHeight; y++)

{

**int** ya = y + yp;

**for**(**int** x = 0; x < eWidth; x++)

{

**int** xa = x + xp;

**if**((xa < 0 || xa >= width) || (ya < 0 || ya >= height)) **break**;

**int** colour = 0;

**int** colourSize = 0;

**if**(entity.getType() == "Tile")

{

Tile tile = entity;

colour = tile.getSprite().getPixels()[x + y \* eWidth];

hiddenColour = tile.getSprite().getHiddenColour();

colourSize = tile.getSprite().getColourSize();

}

**if**(colour != hiddenColour) pixels[xa + ya \* width] = colour;

}

}

}

**void** setOffset(**int** xOffset, **int** yOffset)

{

**this**.xOffset = xOffset;

**this**.yOffset = yOffset;

}

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

}