```
package com.blindtigergames.werescrewed.entity.builders;

import com.badlogic.gdx.graphics.Texture;
import com.badlogic.gdx.math.Rectangle;
import com.badlogic.gdx.math.Vector2;
import com.badlogic.gdx.physics.box2d.BodyDef.BodyType;
import com.badlogic.gdx.physics.box2d.World;
import com.badlogic.gdx.utils.Array;
import com.blindtigergames.werescrewed.WereScrewedGame;
import com.blindtigergames.werescrewed.entity.PolySprite;
import com.blindtigergames.werescrewed.entity.RootSkeleton;
import com.blindtigergames.werescrewed.entity.Skeleton;
import com.blindtigergames.werescrewed.entity.action.FadeSkeletonAction;
import com.blindtigergames.werescrewed.eventTrigger.EventTrigger;

public class SkeletonBuilder extends GenericEntityBuilder< SkeletonBuilder > {

    protected Array< Vector2 > polyVertsFG, polyVertsBG, invisibleVerts;

    protected float density;
    private BodyType bodyType;
    protected boolean onBGverts;
    protected Texture texBackground, texForeground, texBody;
    protected boolean hasDeactivateTrigger;
    protected boolean fadeFgDecals;
    protected boolean setChildSkeletonsToSleep = false;
    protected boolean useBoundingRect = false;
    protected Rectangle boundingRect;

    protected boolean lessExtraBorder = false;

    public SkeletonBuilder( World world ) {
        super( );
        reset( );
        super.world = world;
    }

    @Override
    public SkeletonBuilder reset( ) {
        super.reset( );
        this.polyVertsFG = null;
        this.polyVertsBG = null;
        this.bodyType = BodyType.KinematicBody;
        this.density = 1.0f;
        this.onBGverts = true;
        // background textures
        this.texBackground = WereScrewedGame.manager.getLevelRobotBGTex( );
        this.texForeground = WereScrewedGame.manager.getLevelRobotFGTex( );
        this.texBody = null;
        this.hasDeactivateTrigger = false;
        this.fadeFgDecals = false;
        this.invisibleVerts = null;
        this.setChildSkeletonsToSleep = false;
        return this;
    }

    /**
     * All following verts added will set to the background polysprite of this
     * skeleton This is true by default
     *
     * @return
     */
    public SkeletonBuilder bg( ) {
        this.onBGverts = true;
        return this;
    }

    public SkeletonBuilder lessExtraBorder( ) {
        this.lessExtraBorder = true;
        return this;
    }

    /**
     * All following verts will apply to the foreground polysprite
     *
     * @return
     */
    public SkeletonBuilder fg( ) {
        this.onBGverts = false;
        return this;
    }

    public SkeletonBuilder hasDeactiveTrigger( boolean hasTrigger ) {
        this.hasDeactivateTrigger = hasTrigger;
        return this;
    }

    public SkeletonBuilder texForeground( Texture fgTex ) {
        this.texForeground = fgTex;
        return this;
    }

    public SkeletonBuilder texBackground( Texture bgTex ) {
        this.texBackground = bgTex;
        return this;
    }

    public SkeletonBuilder texBody( Texture bodyTex ) {
        this.texBody = bodyTex;
        return this;
    }

    public SkeletonBuilder setUseBoundingRect( boolean setting ) {
        useBoundingRect = setting;
        return this;
    }

    public SkeletonBuilder buildRectangle( float x, float y, float width, float
height ) {
        boundingRect = new Rectangle( x, y, width, height);
        return this;
    }

    /**
     * Set the entire vertice list for the polySprite on the next built skeleton
     *
     * @param verts
     *            array of verts in pixels.
     * @return
     */
    public SkeletonBuilder setVerts( Array< Vector2 > verts ) {
        if ( onBGverts ) {
            this.polyVertsBG = verts;
        } else {
            this.polyVertsFG = verts;
        }
        return this;
    }

    public SkeletonBuilder invisibleVerts( Array< Vector2 > verts ) {
        this.invisibleVerts = verts;
        return this;
    }

    public SkeletonBuilder setChildSkelsToSleep ( boolean setting ) {
        setChildSkeletonsToSleep = setting;
        return this;
    }

    /**
     * Add a vertice to the polySprite for this skeleton
     *
     * @param vert
     *            , (x,y) in pixels
     * @return
     */
    public SkeletonBuilder vert( Vector2 vert ) {
        Array< Vector2 > vertList;
        if ( onBGverts ) {
            if ( polyVertsBG == null ) {
                polyVertsBG = new Array< Vector2 >( );
            }
            vertList = polyVertsBG;
        } else {
            if ( polyVertsFG == null ) {
                polyVertsFG = new Array< Vector2 >( );
            }
            vertList = polyVertsFG;
        }
        vertList.add( vert );
        return this;
    }

    /**
     * Add a vertice to the polySprite for this skeleton
     *
     * @param x
     *            x-position in pixels
     * @param y
     *            y-position in pixels.
     * @return
     */
    public SkeletonBuilder vert( float x, float y ) {
        return this.vert( new Vector2( x, y ) );
    }

    public SkeletonBuilder dynamic( boolean d ) {
        if ( d ) {
            return this.dynamic( );
        }
        return this.kinematic( );
    }

    public SkeletonBuilder dynamic( ) {
        bodyType = BodyType.DynamicBody;
        return this;
    }

    public SkeletonBuilder staticBody( ) {
        bodyType = BodyType.StaticBody;
        return this;
    }

    public SkeletonBuilder kinematic( ) {
        bodyType = BodyType.KinematicBody;
        return this;
    }

    public SkeletonBuilder fadeFgDecals( boolean applyFadeToFgDecals ) {
        this.fadeFgDecals = applyFadeToFgDecals;
        return this;
    }

    /**
     *
     * @param density
     *            - float used for density, default is 1.0f
     * @return SkeletonBuilder
     */
    public SkeletonBuilder density( float density ) {
        this.density = density;
        return this;
    }



    /**
     * Builds a friggin root skeleton, what do you want jeese.
     */
    public RootSkeleton buildRoot( ) {
        return new RootSkeleton( "root", new Vector2( ), null, world );
    }

    @Override
    public Skeleton build( ) {
        Skeleton out = new Skeleton( name, pos, null, super.world, bodyType );
        out.setChildSkeletonsToSleepProperty( setChildSkeletonsToSleep );
        out.setUseBoundingRect( useBoundingRect );
        out.boundingRect = this.boundingRect;
        if ( invisibleVerts != null ) {
            if ( polyVertsFG != null && texForeground != null ) {
                out.fgSprite = new PolySprite( texForeground, polyVertsFG );
            }
        }
        if ( polyVertsBG != null && texBackground != null ) {
            out.bgSprite = new PolySprite( texBackground, polyVertsBG );
        }

        // out.body.setType( bodyType );
        out.setDensity( this.density );

        if ( invisibleVerts != null ) {
            EventTriggerBuilder etb = new EventTriggerBuilder( world );
            etb.name( name + "-invisible-fader" ).setVerts( invisibleVerts );

            if(this.lessExtraBorder)
                etb.extraBorder( 256f );
            else
                etb.extraBorder( 300f );


            EventTrigger et = etb.position( pos.add( 0, 0 ) ).addEntity( out )
                    .beginAction( new FadeSkeletonAction( true ) )
                    .endAction( new FadeSkeletonAction( false ) ).repeatable( )
                    .twoPlayersToDeactivate( ).build( );

            out.addEventTrigger( et );
        } else {
            // PIZZA
            if ( hasDeactivateTrigger && polyVertsBG != null ) {
                EventTriggerBuilder etb = new EventTriggerBuilder( world );
                etb.name( name + "-activator" ).setVerts( polyVertsBG );


            if(this.lessExtraBorder)
                etb.extraBorder( 128f );
            else
                etb.extraBorder( 300f );

            EventTrigger et = etb.position( pos ).addEntity( out )
```

```
                .beginAction( new FadeSkeletonAction( true ) )
                .endAction( new FadeSkeletonAction( false ) )
                .repeatable( ).twoPlayersToDeactivate( ).build( );
        out.addEventTrigger( et );
            // Gdx.app.log( "SkeletonBuilder",
            // "I just built an event trigger" );
        } else if ( polyVertsFG != null ) {
            EventTriggerBuilder etb = new EventTriggerBuilder( world );

            etb.name( name + "-fg-fader" ).setVerts( polyVertsFG );


        if(this.lessExtraBorder)
            etb.extraBorder( 128f );
        else
            etb.extraBorder( 300f );

            EventTrigger et = etb.position( pos.add( 0, 0 ) ).addEntity( out )
                    .beginAction( new FadeSkeletonAction( true ) )
                    .endAction( new FadeSkeletonAction( false ) )
                    .repeatable( ).twoPlayersToDeactivate( ).build( );
            out.addEventTrigger( et );
        }
    }

    if ( fadeFgDecals ) {
        out.setFgFade( fadeFgDecals );
    }

    return out;
  }
}
```