

31.03.2023 Uldis Baurovskis Armands Baurovskis

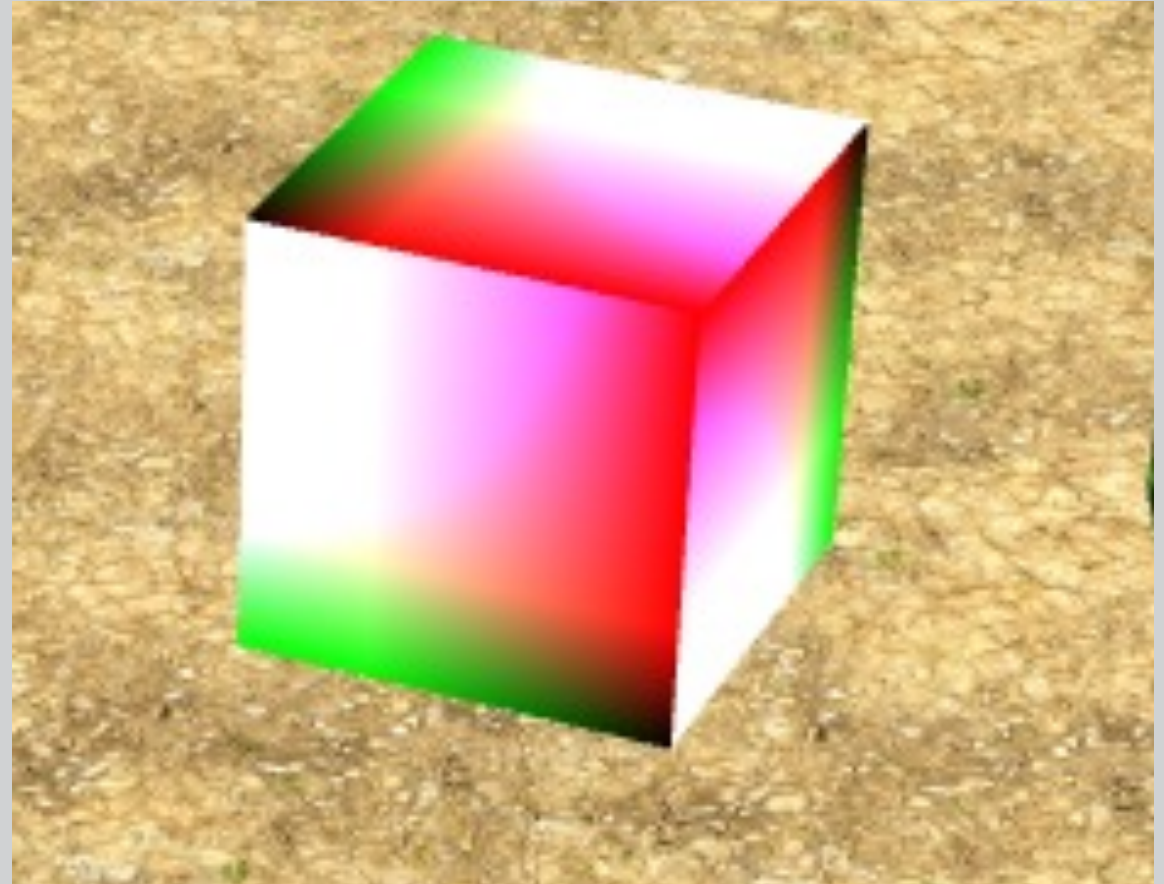
Rat race continous

Agenda

- Power up
- Shaders
- Create custom shader
- Shadow
- Decals
- Normal mapping
- Clone & Instances
- BabylonJS Inspector

Power up

- Boost up player speed
- Made from:
 - Cube as mesh
 - Custom material
 - Intersection for collision



Power up (cont.)

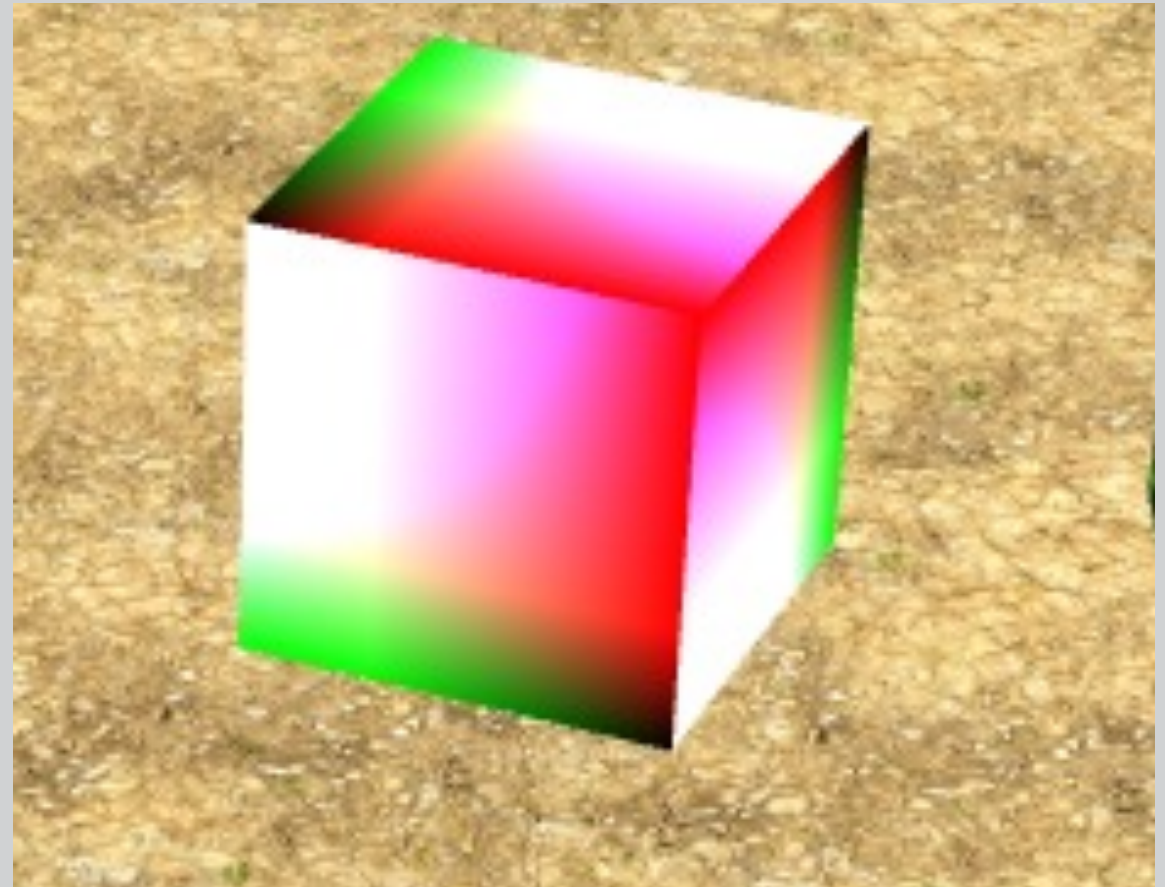
- Shader material:
 - Base for custom materials
 - Allow to define own shader programs

Power up (cont.)

- Shader parts:
 - Vertex shader program
 - Fragment shader program
 - Data types
 - Uniforms and attributes

Power up (cont.)

- Intersection:
 - Way to check if mesh collides with other one
 - Need to be checked on some event (like enter frame)
 - Have 2 modes full check and bounding box check



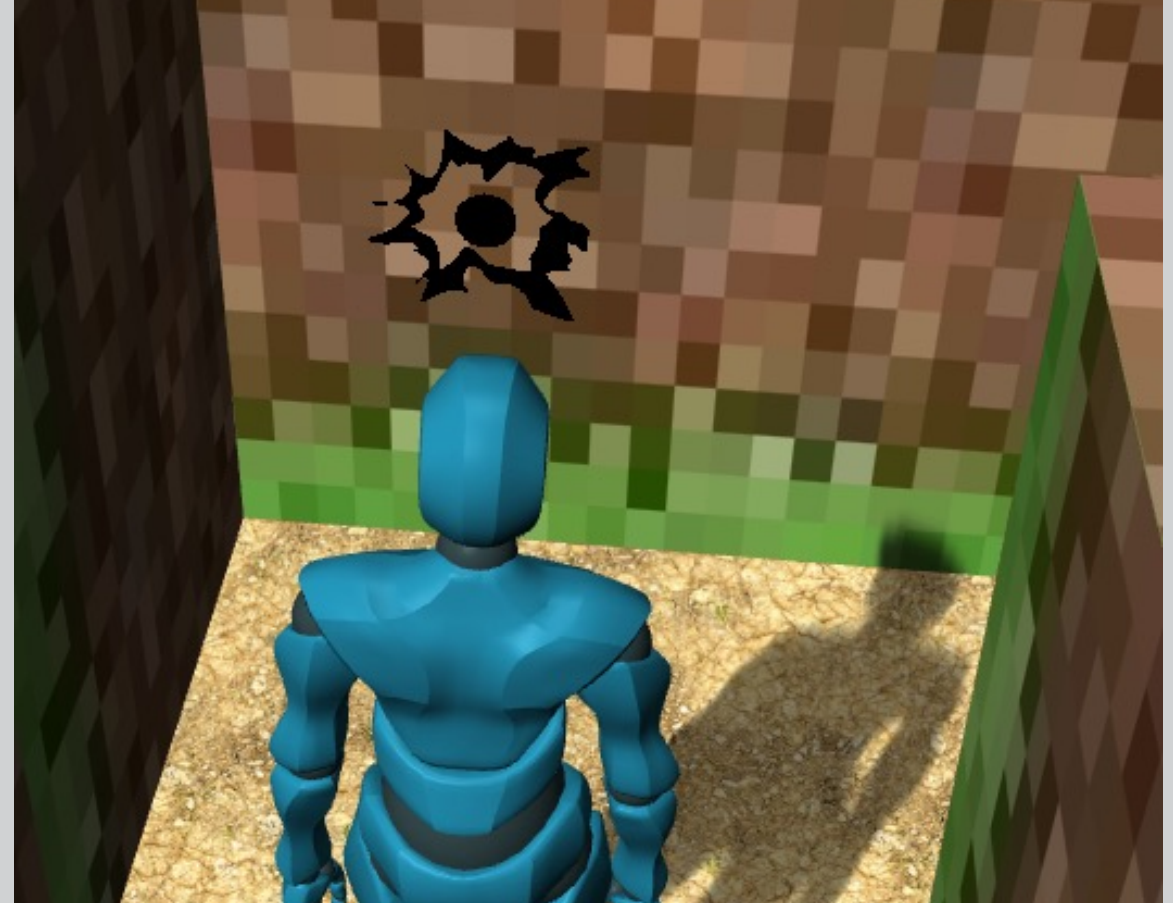
Shadows

- Add shadows for player
- On walls
- Parts:
 - Light that can emit shadows
 - List of shadow receivers
 - List of shadow casters
- Shadow filters



Decals

- Custom image on top of material
- Adds effects to scene
- Parts:
 - Custom decal mesh
 - Decal material
 - Ray casting for position



Decals (cont.)

- Create decal mesh that will be added on top of other mesh
- size – size of decal mesh
- position – position on mesh where decal will be placed
- normal – normals used to deform decal, is necessary to place decal on complex meshes

```
const decal = MeshBuilder.CreateDecal(  
    "decalName"  
    targetMesh,  
    {  
        position: position,  
        normal: normals,  
        size: size  
    }  
);
```

Decals (cont.)

- Need to find a wall using ray casting
- To create a ray, need 3 parts:
 - Origin point
 - Direction vector
 - Distance

```
const decal = MeshBuilder.CreateDecal(  
    "decalName"  
    targetMesh,  
    {  
        position: position,  
        normal: normals,  
        size: size  
    }  
);
```

Decals (cont.)

- Use hit test to check if ray intersects with mesh

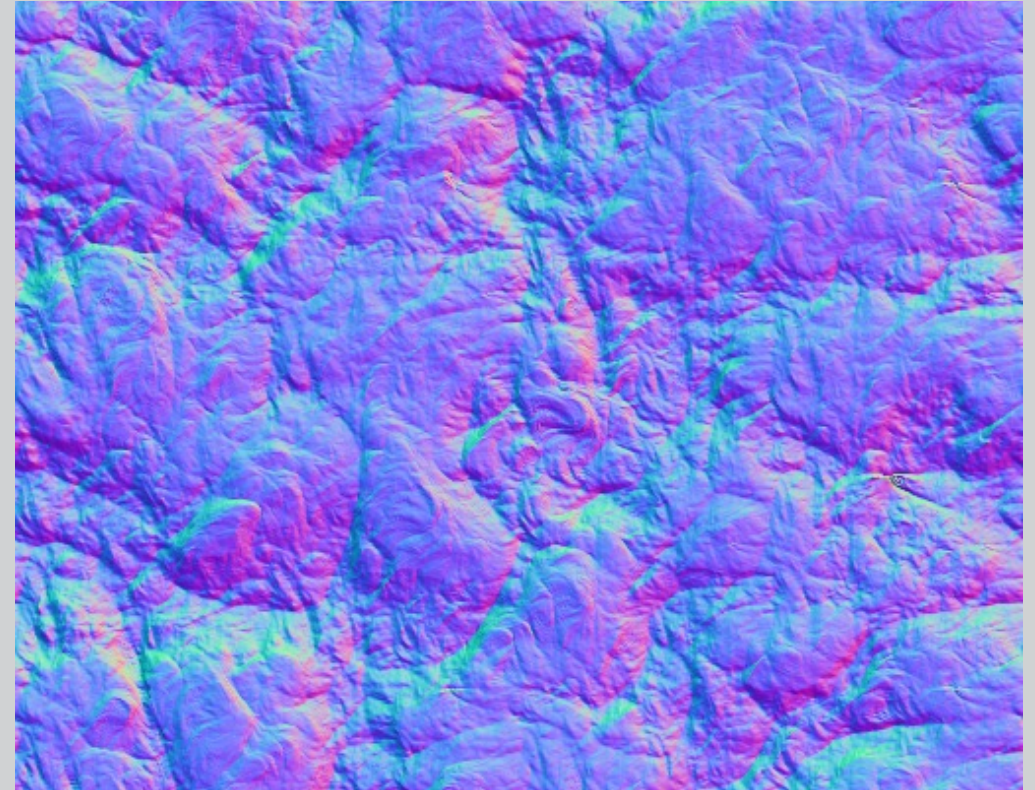
```
const ray = new Ray(origin, direction, rayDistance);

const hit =
this.getScene().pickWithRay(ray, mesh =>
{
    return mesh.name.startsWith("Wall");
});
```

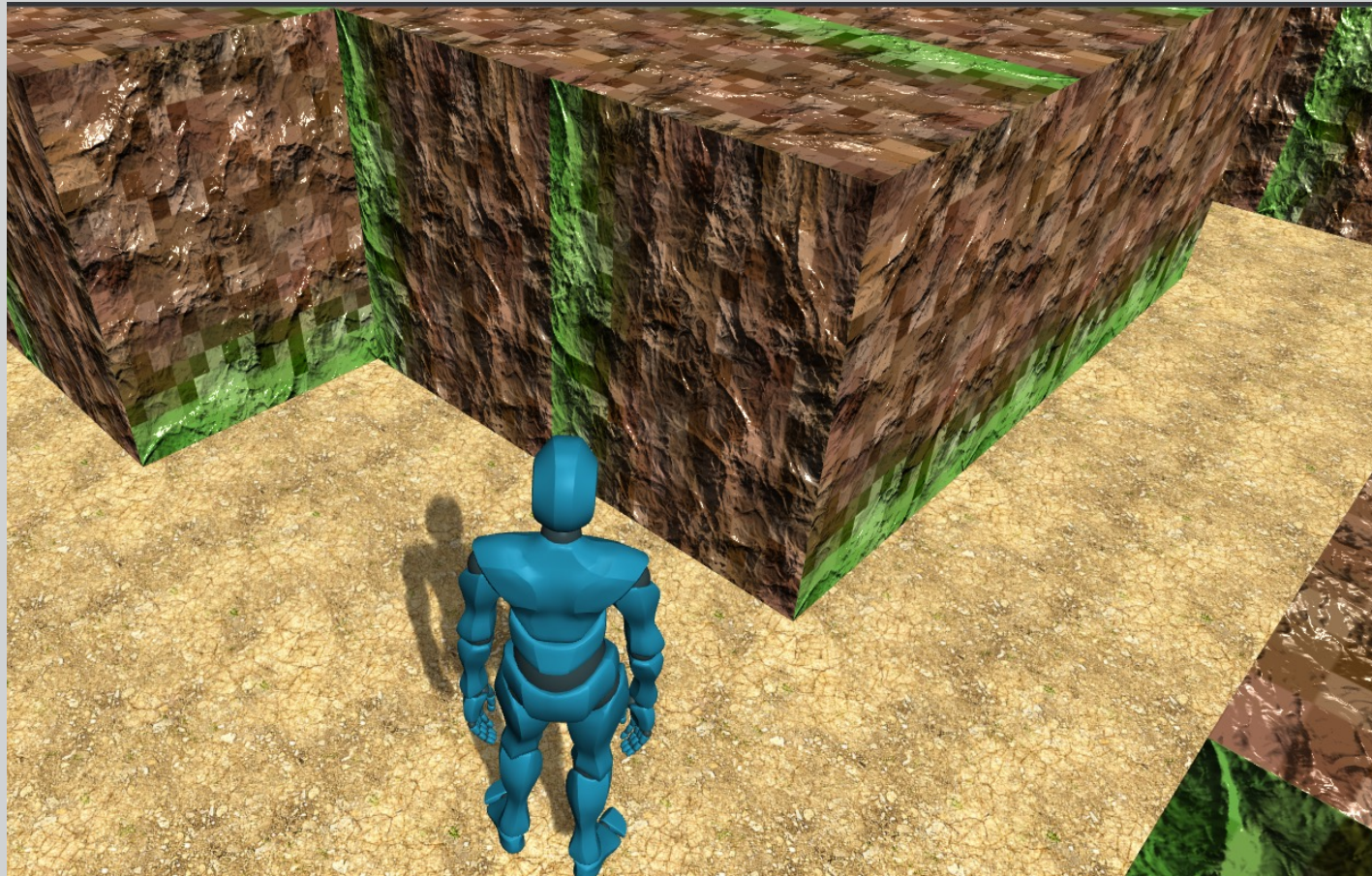
Normal mapping

- Texture mapping technique for faking bumps and dents

```
material.bumpTexture =  
scene.getTextureByName(TextureId.NormalMap);
```



Normal mapping (cont.)



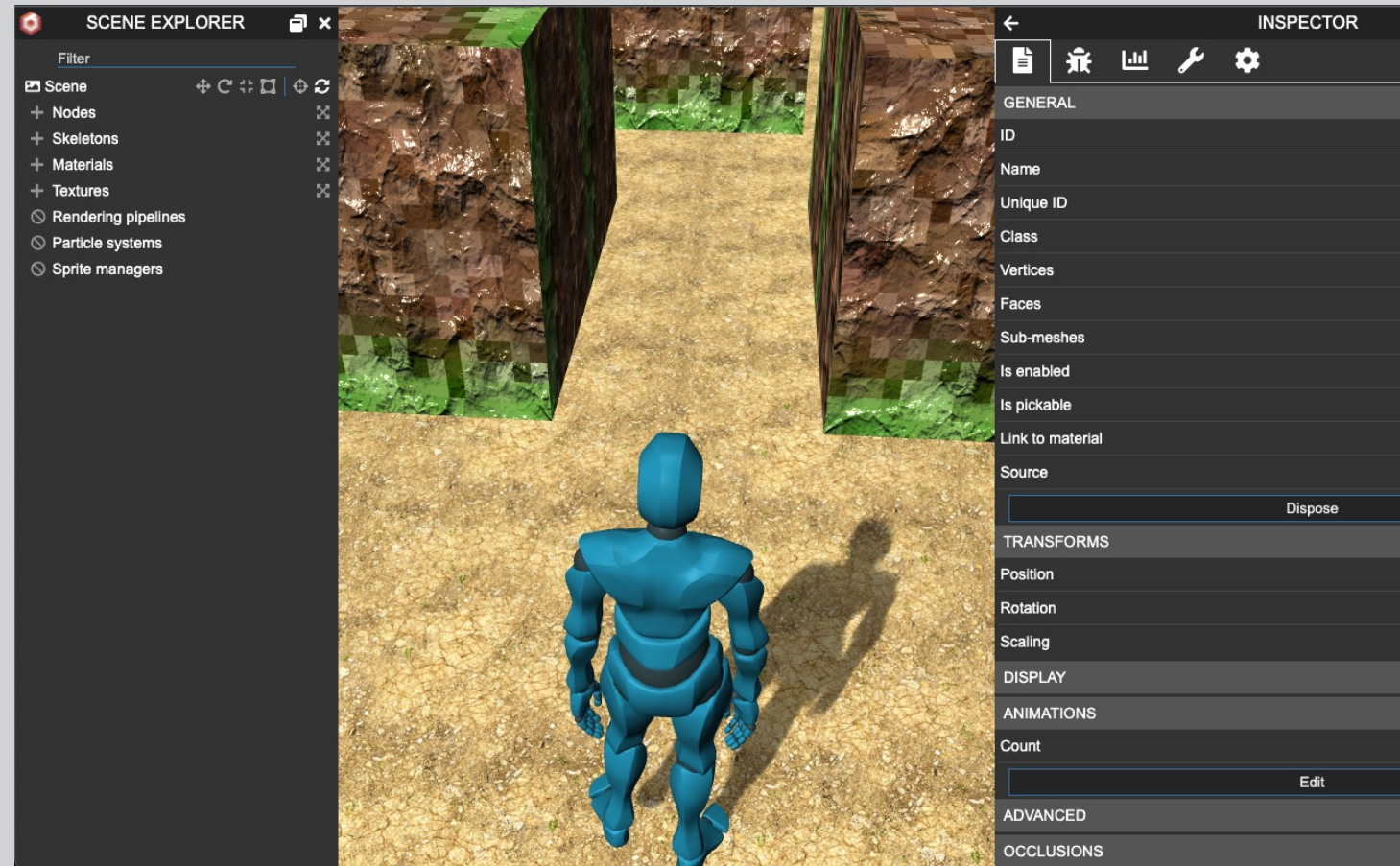
Clone & Instances

- Resource optimization techniques
- Instances:
 - mirrored meshes that can be rendered all together, but share same material so visually are the same
- Clones:
 - Share only geometry so can have individual material per mesh

BabylonJS Inspector

- How to use
- What features contains
- How to debug

```
this.scene.debugLayer.show();
```



Homework

- Try out concepts learend in lecture
- Update game from day 2 with new materials
- Add own powere ups

Q&A