27.03.2023 Uldis Baurovskis Armands Baurovskis

Nuts and bolts of BabylonJS

Agenda

- Intro in Web3D
- Intro in BabylonJS
- Meshes and geometries
- Materials
- Animations
- Lights
- Cameras
- Asset management

Intro in Web3D

- Keywords:
  - WebGL 1 / 2
  - WebGPU
  - Shaders
  - Math (matrices and vectors)

Web3D in action

Intro in BabylonJS

- BabylonJS is one most popular real time 3D engine using JavaScript
- Written in TypeScript
- Rich feature support
- 2013 by David Catuhe, David Rousset
- Now maintained by Microsoft
- Has a lot of great tools:
  - Playground
  - Node material editor

HelloWorld in BabylonJS

- Neccesary parts:
  - Canvas
  - Engine
  - Scene
  - Light
  - Camera
  - Render loop

```
// canvas element that will be in DOM, used to create webgl context
const canvas = document.createElement("canvas");
document.body.appendChild(canvas);

// babylonjs main object
const engine = new Engine(canvas, true, {}, true);

// root for objects created in babylonjs, there can be multiple scenes if needed
const scene = new Scene(engine, {});

// generic light to see objects
const light = new HemisphericLight("light", new Vector3(0, 1, 0), scene);

// defines visible area that is rendered on the canvas
const camera = new ArcRotateCamera("camera",
-Math.PI  * 0.5, Math.PI * 0.25, 12, Vector3.Zero(), scene);

// need to resize engine in order to read canvas size
engine.resize();

// game loop function that is run each frame and draws scene
engine.runRenderLoop(() => {
  scene.render();
});
```

Mesh

- Mesh is container for every visible element in babylonjs
- Mesh includes:
  - Geometry
  - Material

```
const sphere = MeshBuilder.CreateSphere("sphere", { diameter: 1 }, scene);
```

TransformNode

- non visible node used for grouping purposes
- More lightweight then empty mesh as parent

```
const node = new TransformNode(”node",  scene);
```

Hierarchy

- Tree like structure for all nodes (parent / child)
- Sums up scale / position / rotation
- Is used to group nodes in strucutres

```
const parent = MeshBuilder.CreateSphere(”parent", { diameter: 1 }, scene);
const child = MeshBuilder.CreateSphere(”child", { diameter: 1 }, scene);

child.setParent(parent);
```

Material

- Material describes visual look of a mesh
- Can be described using a color and textures
- BabylonJS includes multiple material types
- Is a wrapper for shaders

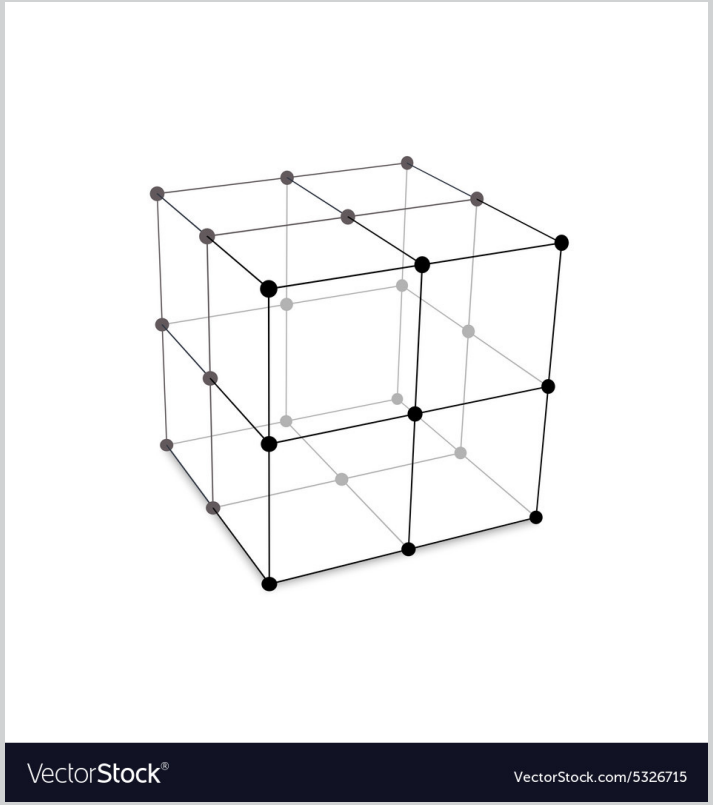
```
const material = new StandardMaterial("material", scene);

material.diffuseColor = new Color3(1,  1,  1);

sphere.material = material;
```

Geometry

- Describe mesh shape using vertexes
- Holds information necessary for gpu to draw shape



Texture

- Texture represents image in babylonjs framework
- Does all the heavy lifting to load image from file and send data to GPU
- Can be static image or canvas
- Textures are used by material to color meshes

```
const texture = new Texture("url", scene);

material.diffuseTexture = texture;
```

Standard material

- Reacts to the light
- Most common properties:
  - Diffuse
  - Emissive
  - Specular
  - Mask
  - Alpha

```
const material = new StandardMaterial("material", scene);

material.diffuseColor = new Color3(1, 1, 1);
material.diffuseTexture = new Texture("diffuse-url", scene);

material.emissiveColor = new Color3(1, 1, 1);
material.emissiveTexture = new Texture("emissive-url", scene);

material.specularColor = new Color3(1, 1, 1);
material.specularPower = 16;
material.specularTexture = new Texture("specular-url", scene);

// mask
material.opacityTexture = new Texture("mask-url", scene);

material.alpha = 0.5;
```

Animations

- Two ways to animate:
  - Using BabylonJS built-in animations
  - Using GameLoop
- Can animate multiple properties:
  - Scale
  - Position
  - Color
  - ….

```
scene.registerBeforeRender(() => {
    sphere.position.x += 0.01 * scene.getAnimationRatio();
});
```

```
const startFrame = 0;
const endFrame = 60;

const animation = new Animation("anim-x", "position.x", 60,
Animation.ANIMATIONTYPE_FLOAT);

const keyFrames = [
  { frame: startFrame, value: 2, },
  { frame: endFrame, value: -2, },
];

animation.setKeys(keyFrames);
sphere.animations.push(animation);
scene.beginAnimation(sphere, startFrame, endFrame, false);
```

Lights

- Is required for seeing mesh volume
- Without lights – mesh are just flat (even cube)
- Used for coloring and shadows
- Most popular types:
  - Directional Light
  - Spot Light
  - Hemispheric Light

```
const directionalLight = new DirectionalLight("light",
 new Vector3(0, -1, 0), scene);

const spotLight = new SpotLight("light", new Vector3(-2, 10, -1),
 new Vector3(0, -1, 0), Math.PI * 0.5, 4, scene);

const hemisphericLight = new HemisphericLight("light",
new Vector3(0, 1, 0), scene);
```

Camera

- Camera is used as term from film industry, but reality is just an more human usable form to set view matrix (visible area).
- In 3d word camera is always static, and world is turning around it
- Has different types:
  - Free
  - ArcRotate
  - Follow

```
const camera = new ArcRotateCamera("camera", -Math.PI * 0.5,
Math.PI * 0.25, 12, Vector3.Zero(), scene);
```

Asset managment

- AssetsManager is used to to load external models/textures asynchronous
- There is multiple ways how to load assets in BabylonJS

```
const assetManager = new AssetsManager(scene);
assetManager.useDefaultLoadingScreen = false;
assetManager.addMeshTask("task-id", "", "url", "");
assetManager
                    .loadAsync()
                    .then(() => {
                        // done with loading
                     })
                    .catch(() => {
                       // error on loading
                    });
```

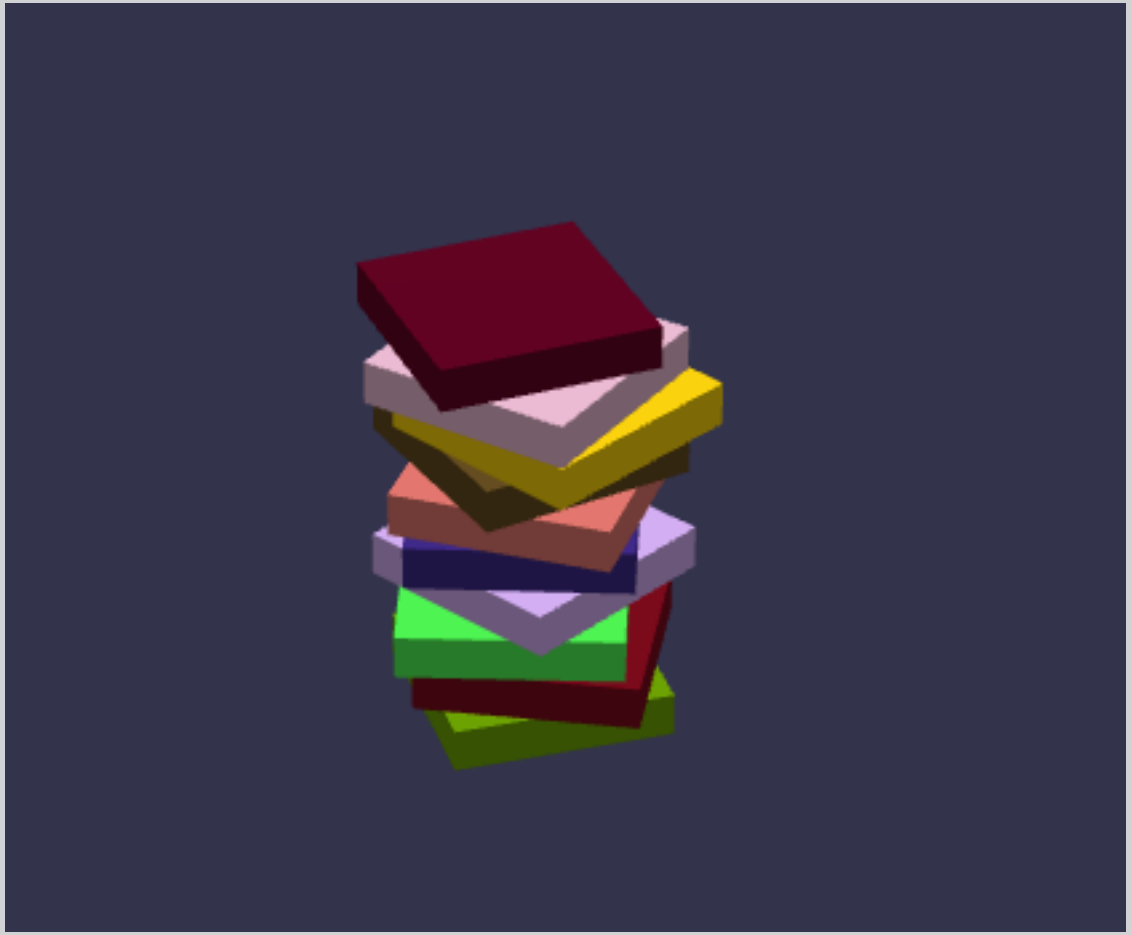
File formats

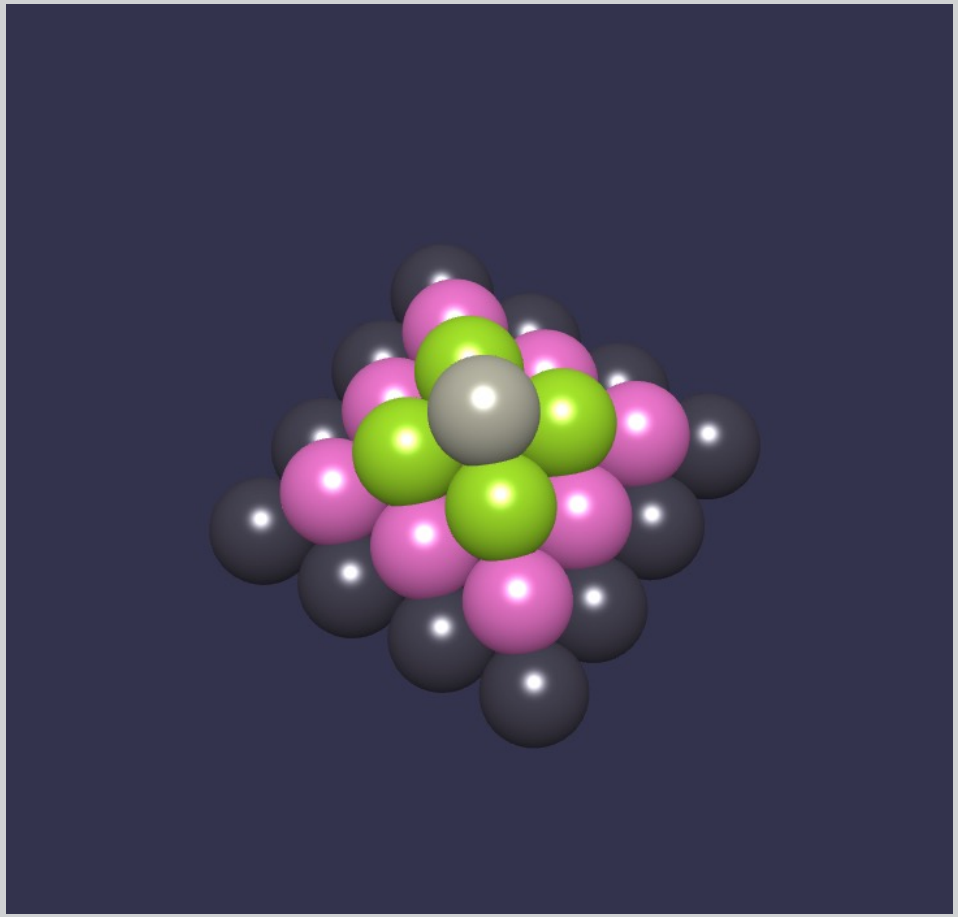
- Most used file formats:
  - Babylon file format
  - glTF
- glTF is global standard file format for models
- To use glTF in BabyblonJS, Loader plugin is needed
- Contains data about models, meshes, materials, geometry

Useful resources

- https://webglfundamentals.org
- https://webgl2fundamentals.org
- https://thebookofshaders.com
- https://www.shadertoy.com
- https://doc.babylonjs.com

Homework





Q&A