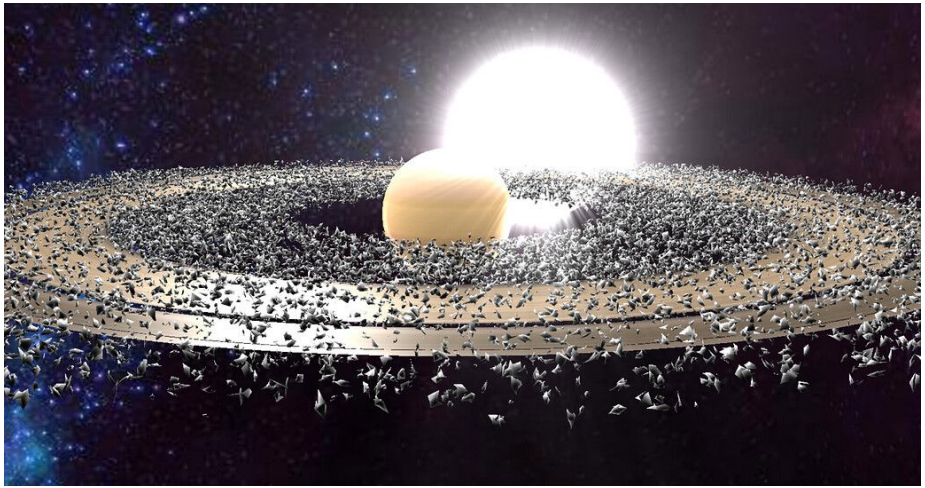
Babylon.js

Vincent Zheng, Daniel Mogilevsky, Kyle Cones, Lezhi Wang, Jin Lin

Learn about

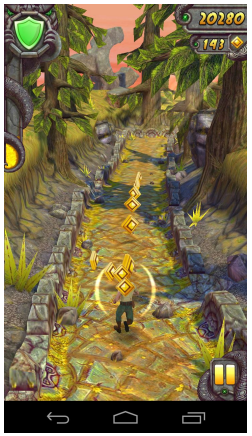
Babylon.js is a Javascript 3D graphics library that can be used to render 3D scenes on web pages.



Uses as a Game Engine

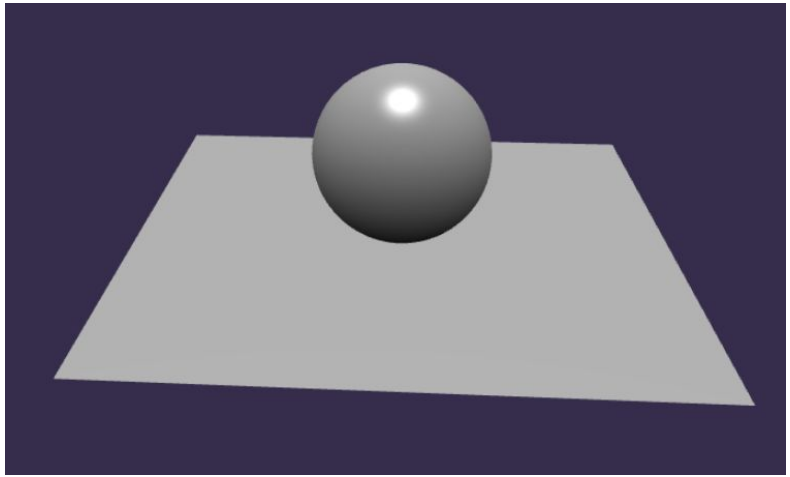
With a built in physics engine and audio support, Babylon.JS doubles as a game engine and has been used to make the several games:

- Minecraft Classic
- Assassins Creed Pirates
- Temple Run 2
- Many more



Features in Babylon.JS

- Various shapes
- Cameras and Lights
- Meshes, textures, and Materials
- Sprites
- Morphing
- Mesh Intersection and collision detection
- Physics engine plug-in
- Action Manager
- Instances and Particles
- Support for Bones and Skeletons
- Adding music and sound to the scene



Engine

Babylon.js have plugin systems for physics engines that enables the the animation of “real-life” physics interactions between scene's objects like collisions.

There are plugins for 4 physics engines, and each engine has its own features and its own way of calculating the body dynamics to emulate realistic physical interaction between objects:

- Cannon.js
- Oimo.js
- Energy.js
- Ammo.js

Example of enabling the Cannon.js physic engine by simply calling the scene's `enablePhysics` function:

```
var scene = new BABYLON.Scene(engine);
var gravityVector = new BABYLON.Vector3(0,-9.81, 0);
var physicsPlugin = new BABYLON.CannonJSPlugin();
scene.enablePhysics(gravityVector, physicsPlugin);
```

Scene

A scene is like the stage where all the contents will be displayed. Different types of mesh objects, cameras and lights are created and placed that make them seen and viewable.

Additional Functionalities:

- Allow for user interaction with scene via keyboard and pointers;
- Customize your displaying screen when your scene is loading;
- optimise your scene to maintain a good speed;
- Renders particular scene to videos or .png files;

Lights

Lights are showcase meshes and specify the brightness and colour of the scene and mesh objects. All meshes allow light to pass through them unless shadow generation attribute is activated. The default number of lights allowed is four but this can be increased.

There are four types of lights that can be used with a range of lighting properties including emissive, diffuse and specular, and ground color which only applies to an Hemispheric Light.

1. The Point Light - think light bulb.
2. The Directional Light - think planet lit by a distant sun.
3. The Spot Light - think of a focused beam of light.
4. The Hemispheric Light - think of the ambient light.

Camera

Babylon.js has different kinds of cameras with the two most used being the universal and arc-rotate cameras

- Universal Camera - Can move freely
- Arc Rotate Camera - acts like a satellite in orbit around a target
- Follow Camera - Chooses a mesh as a target and follows it
- Device Orientation Cameras - Reacts to the tilt of a device, such as a smartphone
- Virtual Reality Camera - a camera for VR devices

Cameras can be attached to the DOM canvas, allowing the user to control them. Example of a controllable universal camera pointing towards the center of the scene:

```
// Parameters : name, position, scene
var camera = new BABYLON.UniversalCamera("UniversalCamera", new BABYLON.Vector3(0, 0, -10),
 scene);

// Targets the camera to a particular position. In this case the scene origin
camera.setTarget(BABYLON.Vector3.Zero());

// Attach the camera to the canvas
camera.attachControl(canvas, true);
```

Meshes

Meshes are collections of small triangles molded together to create a shape.

Babylon.JS allows for the creation of predefined meshes (shapes, which will be talked about later) or making meshes from a set of vertices.

Meshes can have children and parents, this allows for hierarchical manipulation of scenes, as a transformation applied to a parent mesh can recursively be applied to all of its children.

Ex: Mesh hierarchy used to manipulate 3D models in internship

Shapes

Common 2D shapes of any type of regular polygon can be utilized easily.

3D shapes such as cuboids, spheres, ellipsoids, and toruses can also be created with meshes.

A shape has the properties of updatable and instance:

Updatable - When changes are made to the shape defining properties

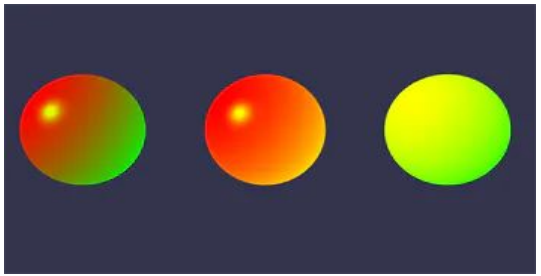
Instance - Re-render the mesh with the updated shape defining properties

Materials

Materials brings color and texture to meshes.

They are able to be displayed as a wireframe, degrees of transparency, blended, or scaled/offset a mesh.

Node materials allow the user to create a material based off of custom shaders but with the easability of not having to handle shader coding.



GUI

The Babylon.js GUI library is an extension you can use to generate interactive user interfaces.

2D GUI's can be built using:

- Buttons
- Sliders
- Controls
- Selectors

And 3D GUI's can be built using:

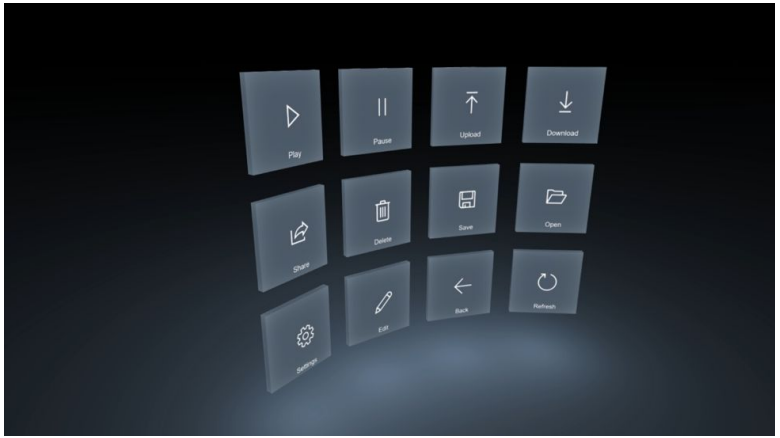
- Holographic Buttons

3D GUI

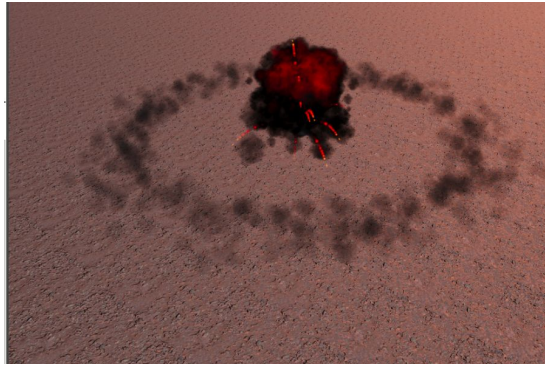
An HolographicButton can be defined with two properties:

text: Gets or sets text for the button

imageUrl: Gets or sets the image url for the button



Particles



Babylon.js has three types of particles based on:

- Sprites
- Meshes
- Points

Solid Particle System:

Can combine multiple meshes into a single mesh which requires a draw call per frame. Many features are available that allow the user to control the individual particles.

Points Cloud Particle System:

An updatable mesh uses a material property to display the vertices as pixel squares. This system also provides a few methods for a user to manage an individual particle.

Interactive Playground (Demo)

Babylon.JS has an interactive playground on their site, we will go there now for a live demo of the different Babylon.JS features

Physics: https://playground.babylonjs.com/#3I55DK#0

Color Picker and VR: https://playground.babylonjs.com/#14KRGG#46

References:

https://www.babylonjs.com/

https://doc.babylonjs.com/