VRSpace.org - Support #58

babylon playground

04/19/2021 11:27 AM - Josip Almasi

Status:	Closed	**Start date:**	04/19/2021
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

make a multiuser playground

History

#1 - 04/19/2021 01:01 PM - Josip Almasi

- Status changed from New to In Progress

Avatar selection:

```
var createScene = function () {
    // This creates a basic Babylon Scene object (non-mesh)
    var scene = new BABYLON.Scene(engine);

    // This creates and positions a free camera (non-mesh)
    var camera = new BABYLON.FreeCamera("camera1", new BABYLON.Vector3(0, 5, -10), scene);

    // This targets the camera to scene origin
    camera.setTarget(BABYLON.Vector3.Zero());

    // This attaches the camera to the canvas
    camera.attachControl(canvas, true);

    import("https://www.vrspace.org/babylon/avatar-selection.js").then( (module) =>{

        var world = module.WORLD;
        world.serverUrl = 'wss://www.vrspace.org/vrspace';
        world.VRSPACEUI.contentBase='https://www.vrspace.org/';

        world.init(engine, 'avatar', scene).then((s) => {
            scene.activeCamera = world.camera;
            world.camera.attachControl(canvas, true);
            world.createSelection((avatar)=>{
                console.log("Selected avatar: "+avatar.getUrl());
            });
            world.showPortals();
        });

    });

    return scene;
};
```

#2 - 04/19/2021 09:06 PM - Josip Almasi

Multiuser world:

```
var createScene = async function () {
    var scene = new BABYLON.Scene(engine);

    import("https://www.vrspace.org/babylon/vrspace-ui.js").then( (ui) =>{

        class PersianCity extends ui.World {
            async createGround() {
                //Ground
                this.ground = BABYLON.Mesh.CreatePlane("ground", 10000.0, this.scene);
                this.ground.material = new BABYLON.StandardMaterial("groundMat", this.scene);
                this.ground.material.diffuseColor = new BABYLON.Color3(.5, 1, .5);
```

```
                this.ground.material.backFaceCulling = false;
                this.ground.material.alpha = 0;
                this.ground.position = new BABYLON.Vector3(-40, 0.4, -20);
                this.ground.rotation = new BABYLON.Vector3(Math.PI / 2, 0, 0);
                this.ground.checkCollisions = true;
            }
            async createSkyBox() {
                var skybox = BABYLON.Mesh.CreateBox("skyBox", 10000, this.scene);
                var skyboxMaterial = new BABYLON.StandardMaterial("skyBox", this.scene);
                skyboxMaterial.backFaceCulling = false;
                skyboxMaterial.disableLighting = true;
                skybox.material = skyboxMaterial;
                skybox.infiniteDistance = true;
                skyboxMaterial.reflectionTexture = new BABYLON.CubeTexture(this.assetPath("../../skybox/hw_sah
ara/sahara"), this.scene);
                skyboxMaterial.reflectionTexture.coordinatesMode = BABYLON.Texture.SKYBOX_MODE;
                return skybox;
            }
            async createCamera() {
                this.camera = this.universalCamera(new BABYLON.Vector3(60, 23, -54));
                this.camera.setTarget(new BABYLON.Vector3(-50,-10,-50));
            }
            async createLights() {
                // Add lights to the scene
                var light = new BABYLON.DirectionalLight("light", new BABYLON.Vector3(-1, -1, 0), this.scene);
                light.intensity = 2;
                var light1 = new BABYLON.HemisphericLight("light1", new BABYLON.Vector3(1, 1, 0), this.scene);
                return light1;
            }

            isSelectableMesh(mesh) {
                return mesh.name == "ground";
            }

            getFloorMeshes() {
                return [ this.ground ];
            }

            setMeshCollisions( mesh, state ) {
                if ( mesh.material && ( mesh.material.id.startsWith("LoamWalls") || mesh.material.id.startsWit
h("Brick") )) {
                    mesh.checkCollisions = state;
                }
            }

            loaded( file, mesh ) {
                mesh.scaling = new BABYLON.Vector3(0.05,0.05,0.05);
                mesh.position.y = .2;
                // TODO FIXME: remove this node from the model
                this.scene.getNodeByID('node4').setEnabled(false);
            }

            registerRenderLoop() {
                // Register a render loop to repeatedly render the scene
                var camera = this.camera;
                this.engine.runRenderLoop(() => {
                    if ( this.terrain && this.terrain.isCreated() ) {
                        //if ( vrHelper.currentVRCamera.position.x > -150 && vrHelper.currentVRCamera.position
.x < 150 && vrHelper.currentVRCamera.position.z >= -150 && vrHelper.currentVRCamera.position.z <= 150 ) {
                        //if ( camera.position.x > -150 && camera.position.x < 150 && camera.position.z >= -15
0 && camera.position.z <= 150 ) {
                        if ( camera.globalPosition.x > -150 && camera.globalPosition.x < 150 && camera.globalP
osition.z >= -150 && camera.globalPosition.z <= 150 ) {
                            this.terrain.enabled(false);
                        } else {
                            this.terrain.enabled(true);
                        }
                    }
                    this.scene.render();
                });
            }

            createTerrain() {
                this.terrainMaterial = new BABYLON.StandardMaterial("terrainMaterial", this.scene)
                var terrainTexture = new BABYLON.Texture(this.assetPath("textures/LoamWalls0012_2_S_1_1_baseCo
```

```
lor.jpeg"), this.scene);
                this.terrainMaterial.ambientTexture = terrainTexture;
                this.terrainMaterial.specularColor = new BABYLON.Color3(0, 0, 0);
                terrainTexture.uScale = 4.0;
                terrainTexture.vScale = terrainTexture.uScale;

                // box to fix ground texture flickering
                var box = BABYLON.MeshBuilder.CreateBox("fixBox", {width:1000, depth:1000,height:.10}, this.sc
ene); // default box
                box.position = new BABYLON.Vector3(-40,0,-20);
                box.material = this.terrainMaterial;
                box.checkCollisions = true;

                import("https://www.vrspace.org/babylon/terrain-desert.js").then((terrain)=>{
                    this.terrain = new terrain.Desert( this, this.terrainMaterial );
                    this.terrain.checkCollisions = true;
                    this.terrain.createTerrain();
                })
            }

        }

        var world = new PersianCity();
        VRSPACEUI.contentBase='https://www.vrspace.org/';
        world.baseUrl = 'https://www.vrspace.org/content/worlds/persian-city/';
        world.init(engine, 'persian-city', scene);
        world.serverUrl = 'wss://www.vrspace.org/vrspace';

        // create world manager
        var manager = new ui.WorldManager(world);
        manager.debug = true; // world debug to console
        manager.VRSPACE.debug = true; // network debug to console

        // all set, now connect to vrspace.org server with your avatar
        manager.enter( {mesh:"https://www.vrspace.org/babylon/dolphin.glb"} );

    });

    return scene;
};
```

#3 - 04/24/2021 01:05 PM - Josip Almasi

Basic multi-user interaction:

```
var obj1 = null;
var obj2 = null;

var createScene = async function () {
    // import the API, then create the world
    await import('//www.vrspace.org/babylon/vrspace-ui.js').then( (ui) => {
        class Connect extends ui.World {
            async createScene() {
                this.engine = engine;
                this.scene = new BABYLON.Scene(engine)
                this.camera = new BABYLON.UniversalCamera("UniversalCamera", new BABYLON.Vector3(0, 2, -10), t
his.scene);
                this.camera.maxZ = 100000;
                this.camera.setTarget(BABYLON.Vector3.Zero());
                this.camera.attachControl( canvas );
                this.camera.checkCollisions = true;
                var light1 = new BABYLON.HemisphericLight("light1", new BABYLON.Vector3(1, 1, 0), this.scene);
                var light2 = new BABYLON.PointLight("light2", new BABYLON.Vector3(1, 3, -3), this.scene);
                var skybox = BABYLON.Mesh.CreateBox("skyBox", 100.0, this.scene);
                var skyboxMaterial = new BABYLON.StandardMaterial("skyBox", this.scene);
                skyboxMaterial.backFaceCulling = false;
                skyboxMaterial.disableLighting = true;
                skybox.material = skyboxMaterial;
                skybox.infiniteDistance = true;
                skyboxMaterial.reflectionTexture = new BABYLON.CubeTexture("//www.vrspace.org/content/skybox/e
so_milkyway/milkyway", this.scene);
                skyboxMaterial.reflectionTexture.coordinatesMode = BABYLON.Texture.SKYBOX_MODE;
            }
        }
```

```
        var world = new Connect();
        world.createScene();
        world.registerRenderLoop();

        var touch1 = BABYLON.MeshBuilder.CreateSphere("touch1", {diameter:0.5}, world.scene);
        touch1.position = new BABYLON.Vector3(-0.3,0,-0.4);
        var touch2 = BABYLON.MeshBuilder.CreateSphere("touch2", {diameter:0.5}, world.scene);
        touch2.position = new BABYLON.Vector3(0.3,0,-0.4);

        // create world manager to handle shared users, objects and events
        var net = new ui.WorldManager(world);
        net.debug = true; // multi-user debug info
        net.VRSPACE.debug = true; // network debug info

        // all users in the same world share same events
        world.name = 'my very own unique world name';
        // server url defaults to the server serving the script
        // everyone can connect to this one:
        world.serverUrl = 'wss://www.vrspace.org/vrspace';

        // connect, set own avatar and start the session
        net.enter({mesh:'//www.vrspace.org/babylon/dolphin.glb'});

        world.scene.onPointerPick = (e,p) => {
            console.log("Picked ", p.pickedMesh);

            if ( p.pickedMesh === touch1 ) {
                // create a VRObject with custom properties
                if ( ! obj1 ) {
                    net.VRSPACE.createSharedObject({
                    mesh:'//www.vrspace.org/babylon/dolphin.glb',
                    position:{x:touch1.position.x, y:touch1.position.y, z:touch1.position.z},
                    properties:{ stringProperty:'whatever', numberProperty:123.45},
                    active:true
                    }, (obj)=>{
                    console.log("Created new VRObject", obj);
                    // obj1 = obj; // see addSceneListener below
                    });
                } else {
                    net.VRSPACE.deleteSharedObject(obj1);
                    obj1 = null;
                }
                return;
            }

            if ( p.pickedMesh === touch2 ) {
                if ( !obj2 ) {
                    // use a known class to create an object
                    var o = new net.VRSPACE.Client();
                    o.mesh = '//www.vrspace.org/babylon/dolphin.glb';
                    o.active = true;
                    o.name = 'My test client '+Date.now(); // must be unique
                    o.position = {x:touch2.position.x, y:touch2.position.y, z:touch2.position.z};
                    net.VRSPACE.createSharedObject(o, (obj)=>{
                    console.log("Shared object", obj);
                    // no custom event handling, no shared touching
                    obj2 = obj;
                    });
                } else {
                    net.VRSPACE.deleteSharedObject(obj2);
                    obj2 = null;
                }
                return;
            }

            var pickedRoot = ui.VRSPACEUI.findRootNode( p.pickedMesh );
            // root contains VRObject - a shared object
            if ( pickedRoot.VRObject ) {
            console.log('picked shared object ', pickedRoot.VRObject);
            var animationGroup = pickedRoot.VRObject.container.animationGroups[0];
            if ( obj1 && pickedRoot.VRObject == obj1 ) {
                // send a custom event - desired animation state
                net.VRSPACE.sendEvent(obj1, {myEvent:!animationGroup.isPlaying});
            }
            if ( obj2 && pickedRoot.VRObject == obj2 ) {
```

```
                // send a standard event - object position
                net.VRSPACE.sendEvent(obj2, {position: {x:obj2.position.x, y:obj2.position.y+.1, z:obj2.positi
on.z}});
            }
            }

        }

        // this gets triggers whenever any client receives any new VRobject
        net.VRSPACE.addSceneListener( (sceneEvent) => {
            // identify the object
            if ( sceneEvent.added && sceneEvent.added.properties && sceneEvent.added.properties.stringProperty
 == 'whatever') {
                // and NOW install event handler
                // (obviously, installing/overriding the handler at class level makes it easier)
                sceneEvent.added.myEventChanged = (o) => {
                    console.log("Performing remote change: "+o.myEvent);
                    var animationGroup = o.container.animationGroups[0];
                    if ( o.myEvent ) {
                        animationGroup.play(true);
                    } else {
                        animationGroup.stop();
                    }
                };
                // keep the reference, share the event when touched on
                obj1 = sceneEvent.added;
            }
        });

        return world.scene;
    });

};
```

#4 - 04/24/2021 05:34 PM - Josip Almasi

multiuser test: https://playground.babylonjs.com/#ZBK155
persia: https://playground.babylonjs.com/#Y6ILJ5
portal: https://playground.babylonjs.com/#HDV7LA

#5 - 04/24/2021 06:27 PM - Josip Almasi

- Status changed from In Progress to Resolved

https://forum.babylonjs.com/t/multi-user-playgrounds/20288

#6 - 05/22/2021 08:32 PM - Josip Almasi

Template world:

```
var createScene = async function () {
    var scene = new BABYLON.Scene(engine);

    import("https://www.vrspace.org/babylon/vrspace-ui.js").then( (ui) =>{

        class WorldTemplate extends ui.World {
            // OPTIONAL:
            // specify world file name and location
            constructor() {
                super();
                // your world file name, defaults to scene.gltf
                this.file='dolphin.glb';
                // your world directory, defaults to location of world script
                this.baseUrl='//www.vrspace.org/babylon/';
            }
            // MANDATORY: you must create at least one camera
            async createCamera() {
                // utility function to create UniversalCamera:
                this.camera = this.universalCamera(new BABYLON.Vector3(0, 2, -10));
                this.camera.setTarget(new BABYLON.Vector3(0,2,0));
                this.camera.speed = .2;
                return this.camera;
            }

            // OPTIONAL, RECOMMENDED:
```

```
            // start with ground mesh enabled and visible, for reference
            async createGround() {
                this.ground = BABYLON.MeshBuilder.CreateDisc("ground", {radius:100}, this.scene);
                this.ground.rotation = new BABYLON.Vector3( Math.PI/2, 0, 0 );
                this.ground.position = new BABYLON.Vector3( 0, -0.05, 0 );
                this.ground.parent = this.floorGroup;
                //this.ground.isVisible = false;
                this.ground.checkCollisions = true;

                // handy function for dynamic script loading
                await VRSPACEUI.loadScriptsToDocument([
                    "//www.vrspace.org/babylon/babylon.gridMaterial.min.js"
                ]);
                // handy material
                this.ground.material = new BABYLON.GridMaterial("groundMaterial", this.scene);
                this.ground.material.opacity = 0.95;
                return this.ground;
            }

            // OPTIONAL, RECOMMENDED:
            // make some light(s), return one
            async createLights() {
                var light = new BABYLON.DirectionalLight("light", new BABYLON.Vector3(-1, -1, 0), this.scene);
                light.intensity = 2;
                var light1 = new BABYLON.HemisphericLight("light1", new BABYLON.Vector3(0, 1, 0), this.scene);
                return light1;
            }

            // OPTIONAL:
            // create a skybox
            async createSkyBox() {
                var skybox = BABYLON.Mesh.CreateBox("skyBox", 10000, this.scene);
                var skyboxMaterial = new BABYLON.StandardMaterial("skyBox", this.scene);
                skyboxMaterial.backFaceCulling = false;
                skyboxMaterial.disableLighting = true;
                skybox.material = skyboxMaterial;
                skybox.infiniteDistance = true;
                skyboxMaterial.reflectionTexture = new BABYLON.CubeTexture("//www.babylonjs.com/assets/skybox/
TropicalSunnyDay", this.scene);
                skyboxMaterial.reflectionTexture.coordinatesMode = BABYLON.Texture.SKYBOX_MODE;
                return skybox;
            }

            // OPTIONAL, RECOMMENDED:
            createPhysics() {
                // default Earth gravity is too high, set your own here
                this.scene.gravity = new BABYLON.Vector3(0,-.05,0);
            }

            // OPTIONAL, RECOMMENDED:
            // executed once the world is loaded
            loaded(file, mesh) {
                super.loaded(file, mesh);
                // position, rotate, scale and otherwise manipulate world mesh here
                mesh.scaling = new BABYLON.Vector3(2,2,2);
                mesh.rotation = new BABYLON.Vector3(0,Math.PI,0);
            }

            // OPTIONAL, RECOMMENDED:
            // executed once connected to the server and entered the space
            entered( welcome ) {
                // welcome contains client object, but it can always also be accessed from elsewhere
                // at this point this.worldManager is available
                console.log("CONNECTED as "+welcome.client.id, this.worldManager.VRSPACE.me);
            }

        }

        var world = new WorldTemplate();
        VRSPACEUI.contentBase='https://www.vrspace.org/';

        // MANDATORY: initalize
        // RECOMMENDED: set world name
        // all users in the same world share same events
        world.init(engine, 'CHANGEME', this.scene);
```

```
        world.serverUrl = 'wss://www.vrspace.org/vrspace';

        // create world manager
        var manager = new ui.WorldManager(world);

        // OPTIONAL set your own framerate (default is 5)
        manager.fps = 25;
        // with 25 fps no need to create animation in between events
        manager.createAnimations = false;

        // OPTIONAL print debug info to console
        // (a lot of debug with 25 fps)
        //manager.debug = true; // multi-user debug info
        //manager.VRSPACE.debug = true; // network debug info

        // connect, set own avatar and start the session
        manager.enter({
            // RECOMMENDED set your avatar url, or be invisible
            mesh:'//www.vrspace.org/content/char/male/bruce_lee/scene.gltf',
            // OPTIONAL set your name, MUST be unique
            name:'someone'+Math.random()
        });

    });

    return scene;
};
```

#7 - 05/28/2021 09:21 PM - Josip Almasi

- Status changed from Resolved to Closed