CEDEC2016
Computer Entertainment Developers Conference

PLAYCANVAS

新卒1年目でもできる！
クラウド型ゲームエンジン「PlayCanvas」ワークショップ

GMOクラウド株式会社
ソリューション事業部
PlayCanvas運営事務局
テクニカルアドバイザー　津田良太郎

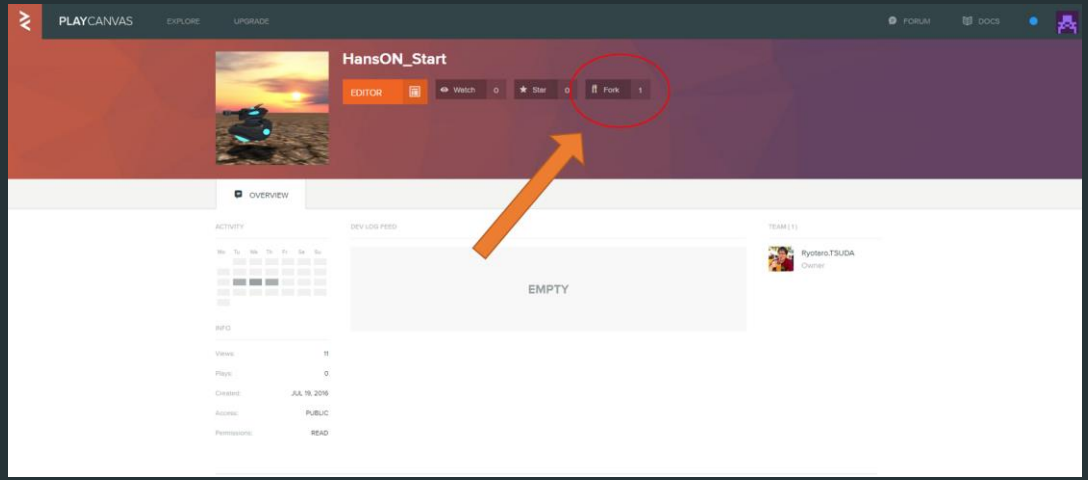
1. プロジェクトの作成

まずはプロジェクトを作成しましょう。

本ワークショップではあらかじめスタートのプロジェクトを用意していますので、それをフォーク(複製)して始めます。

https://playcanvas.com/project/415185/overview/hanson_start にアクセスしてプロジェクトをフォークしてください。

フォークするとPROJECT NAMEが聞かれるので、好きな名前を入力してFORKしてください。

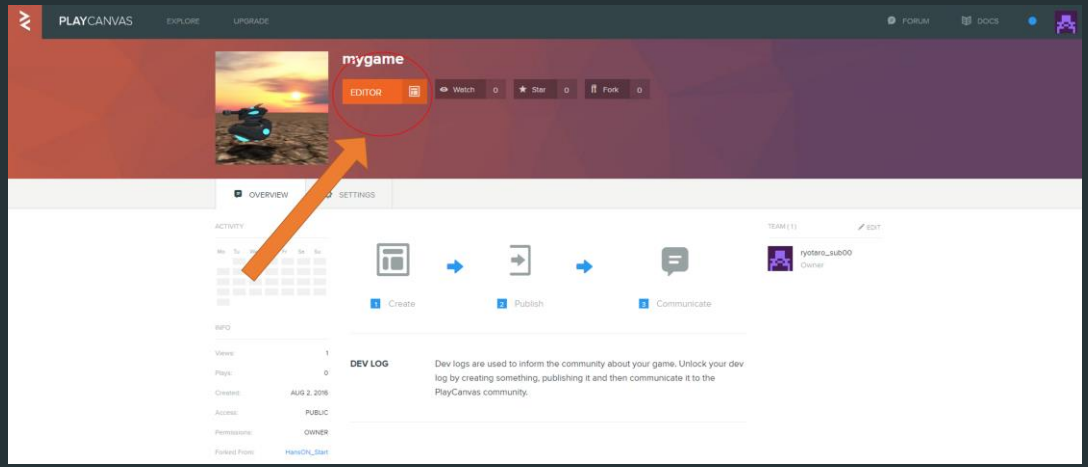


2. プロジェクトの作成

フォークが終わると自分のプロジェクトに登録されます。

これでプロジェクトの作成は終了です。

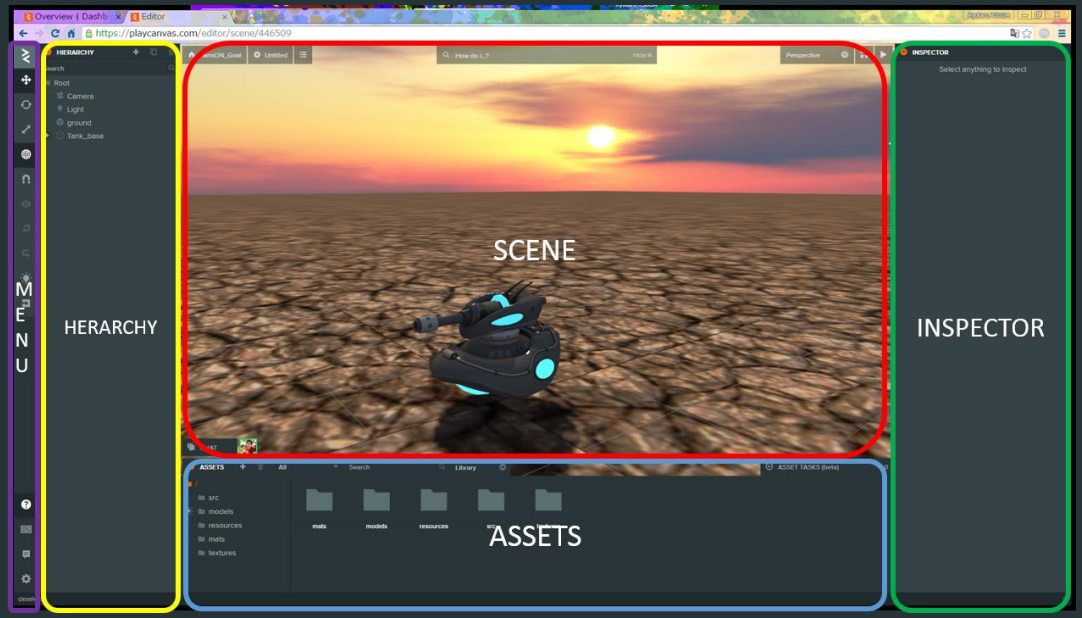
登録されたらEDITORを押して、EDITORを開きましょう。



3. PlayCanvas Editor

Editorを起動すると右のような画面になります。
PlayCanvasはスクリプト作成以外の作業をこのEditorから操作することができます。
Editorの構成は右図のようになっています。

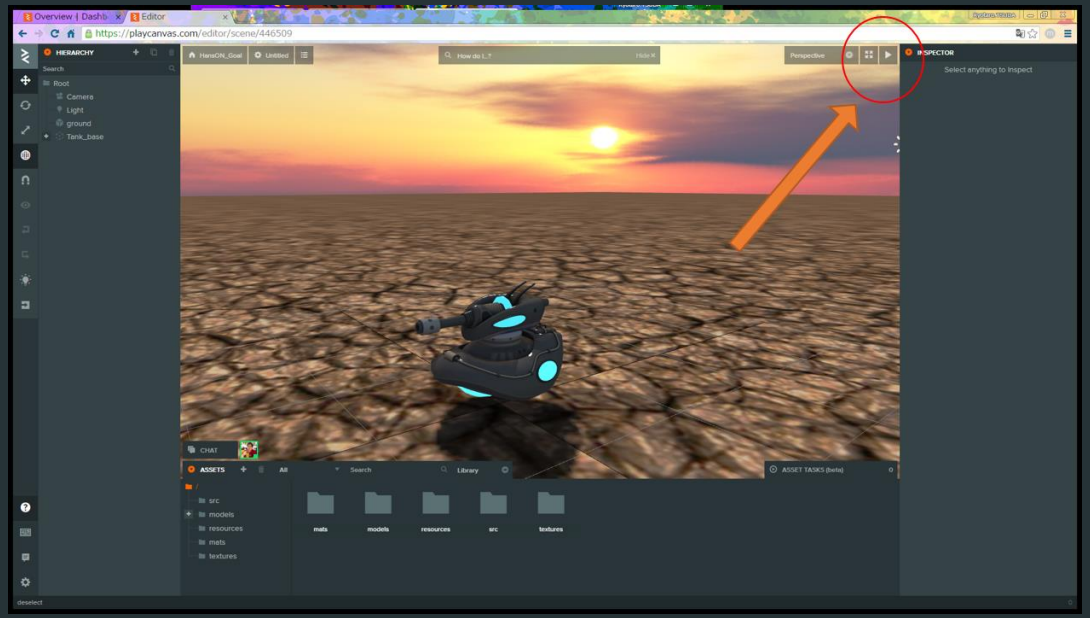
1. シーン(SCENE)
シーンビューには製作中のゲーム世界（シーン）が表示され、自由な位置・角度から眺めることができます。
2. インスペクター(INSPECTOR)
シーンの中で選択肢中のオブジェクトが持つ属性を表示・編集するためのビューです。 属性には座標やメッシュといった見た目上のものから、衝突判定や物理制御に関するパラメーターなどもあり、その他ユーザー定義のものもここに表示されます。
3. ヒエラルキー(HIERARCHY)
シーン内に存在するオブジェクトの一覧が表示されます。 編集中のシーン内でオブジェクトをコピー/ペーストしたり、適切な名前をつけて整理することもできます。
4. アセット(ASSETS)
製作中のプロジェクト（ゲーム全体）に含まれるモデル、スクリプト、グラフィックやサウンドなどのデータ、その他のリソースがファイル単位で表示されます。
5. メニュー(MENU)
シーンのビューモードやプロジェクトセッティング等の作業が行えます。



4. ゲームの実行

ゲームを作り始める前に、現状の機能を確認しておきましょう。PlayCanvasは「ゲーム再生ボタン」をクリックすることで、ゲームの挙動を確認することが出来ます。

ゲームを再生して下さい。ゲームの再生は別タブのlaunchで実行されます。



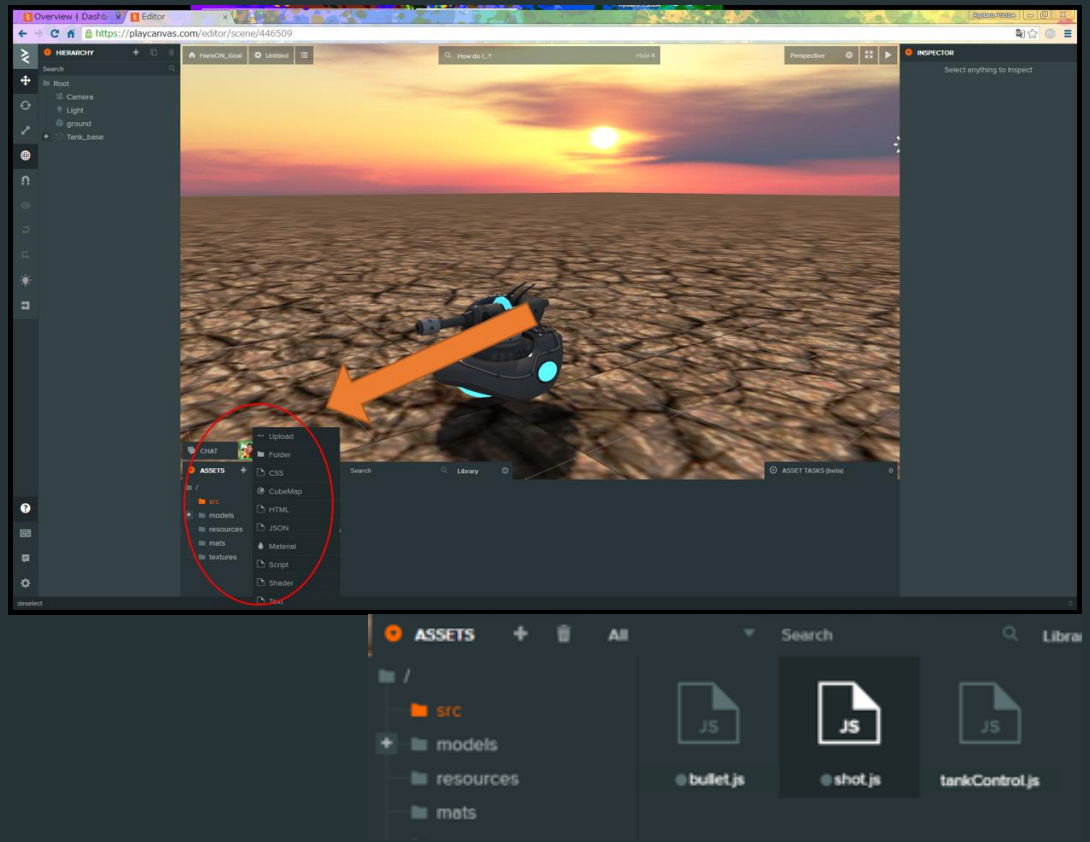
5. ユーザー入力を受け取る

スペースキーを入力したら戦車が弾を発射するような処理を追加しましょう。

まず最初にユーザー入力と弾の発射を制御するスクリプト、弾の挙動を制御するスクリプトをそれぞれ作成します。

ASSETS内の「src」フォルダを選択し、右側の＋からScriptを新規作成します。

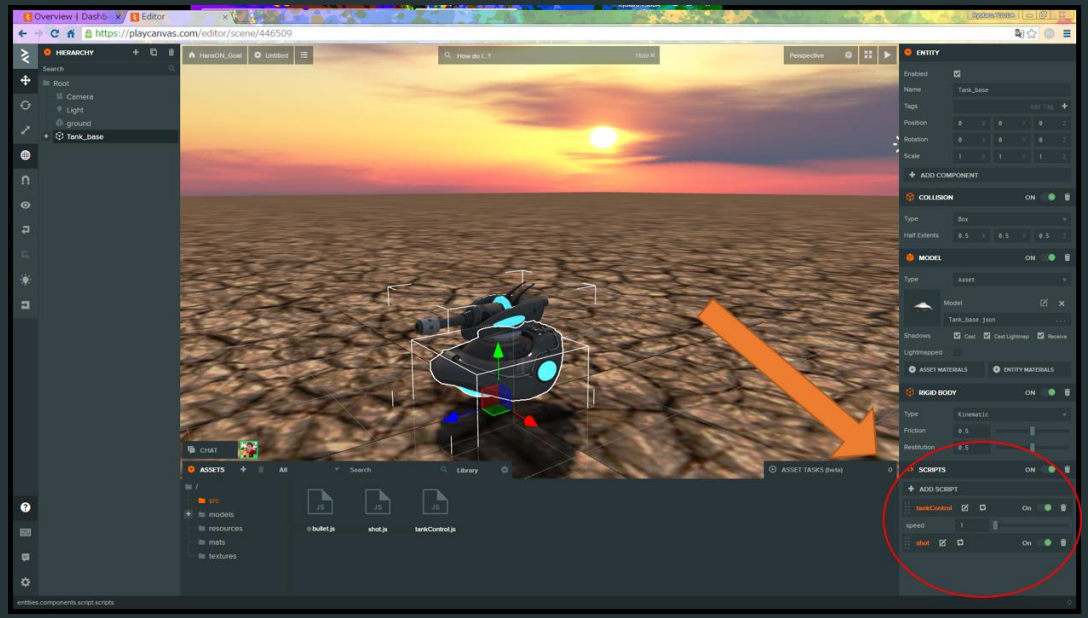
[shot.js], [bullet.js]をそれぞれ作成してください。（※「.js」は自動で付与されます。）



6. Scriptの追加

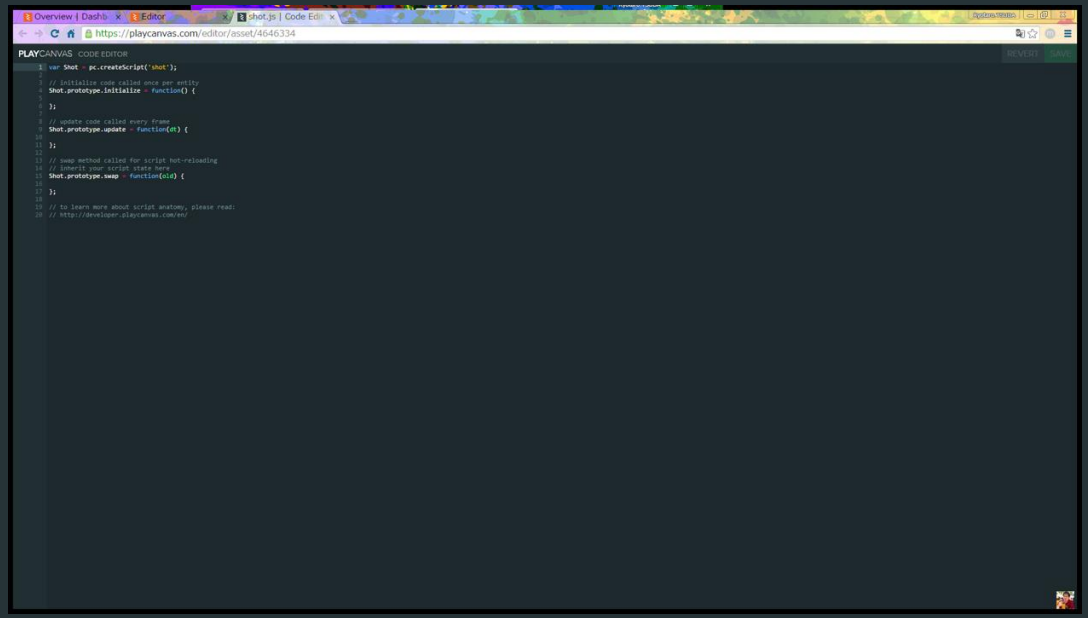
弾は戦車から発射されるため、shot.jsを戦車のコンポーネント（部品）に追加します。

HIERARCHYから「Tank_base」を選択して、INSPECTOR内のSCRIPTSコンポーネント内のADD SCRIPTから「shot.js」を追加します。



7. Scriptの追加

「shot.js」が追加されたことが確認できたら、Editを選択してCode Editorを起動します。



スクリプトには標準で3つのメソッドが実装されています。

initializeメソッド…entityひとつにつき一度だけ実行されるメソッド

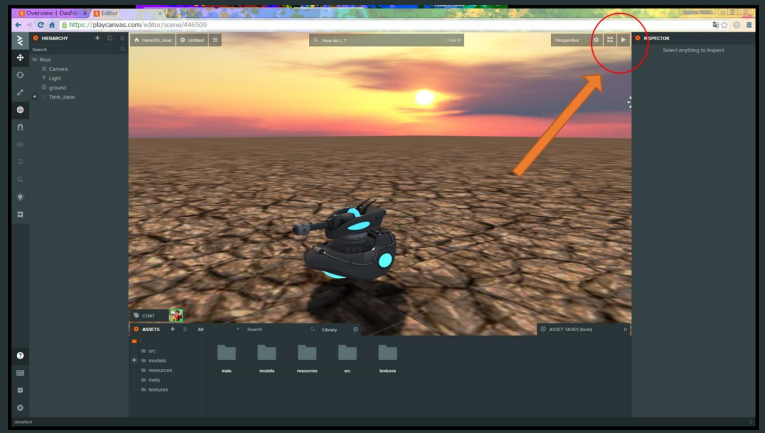
updateメソッド…毎フレーム実行されるメソッド

swapメソッド…hot-reloading時に実行されるメソッド

8. ユーザー入力を受け取る

shot.jsはユーザー入力を受け取り弾を発射するスクリプトなので右のように記述します。

弾はまだ準備していないのでここではconsole.logに"shot!"と出力するようにします。
保存したらGameEditorに戻り、実行してみましょう。



shot.js

```
var Shot = pc.createScript('shot');

// initialize code called once per entity
Shot.prototype.initialize = function() {

};

// update code called every frame
Shot.prototype.update = function(dt) {
    if(this.app.keyboard.wasPressed(pc.KEY_SPACE)){
        //Spaceキーを押したら
        this.shot();
    }
};

// swap method called for script hot-reloading
// inherit your script state here
Shot.prototype.swap = function(old) {

};

Shot.prototype.shot = function(){
       //弾を発射する
       console.log("shot!");
};

// to learn more about script anatomy, please read:
// http://developer.playcanvas.com/en/
```

9. ユーザー入力を受け取る

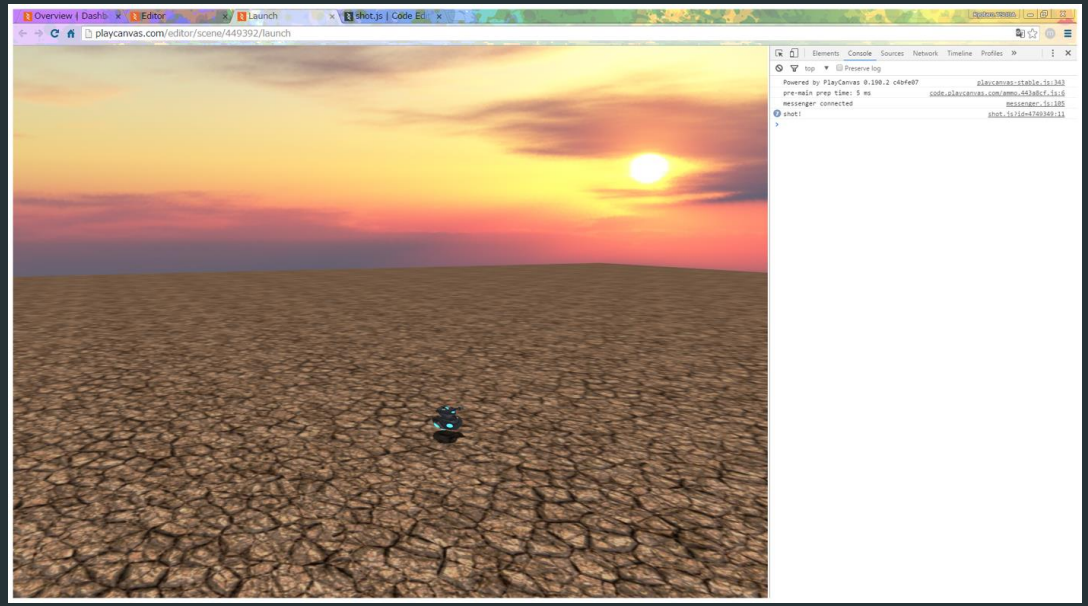
launchが立ち上がったら、ブラウザのデベロッパーツールを起動しましょう。

Windowsの方は「F12」キー

Macの方は「command + option + I」キーを押すと起動します。

デベロッパーツールが起動したら、consoleタブを開きます。

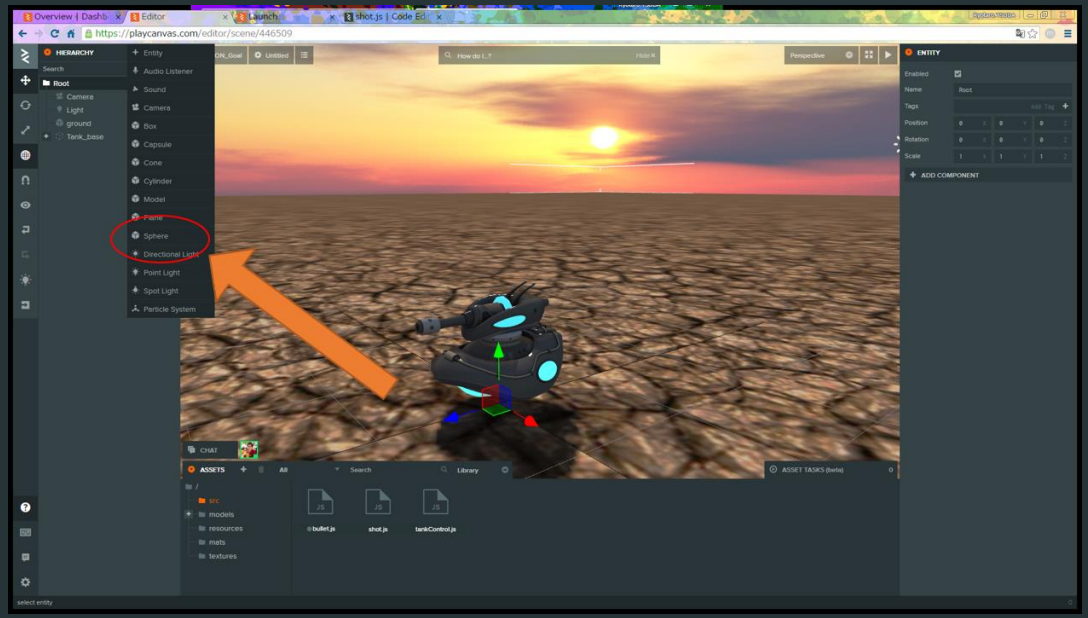
スペースキーを押すとconsoleに"shot!"と出力されることを確認します。



10. 弾のEntityを作成する

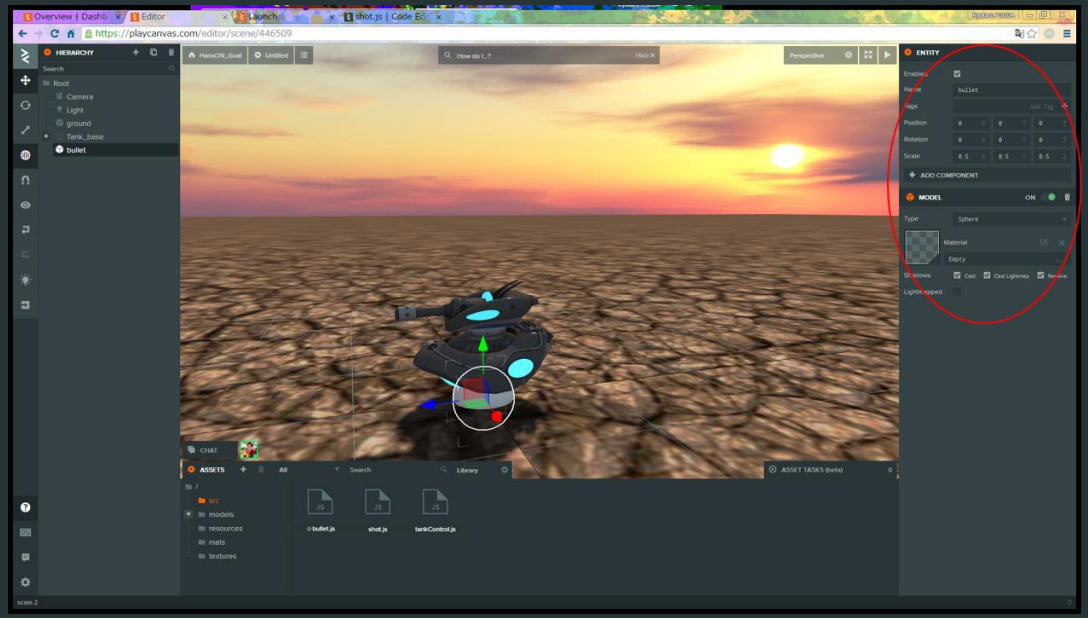
さて、次は発射する弾を作っていきましょう。

HIERARCHYのRootを選択して、右の+からRoot直下にSphereを作成します。



11. 弾のEntityを作成する

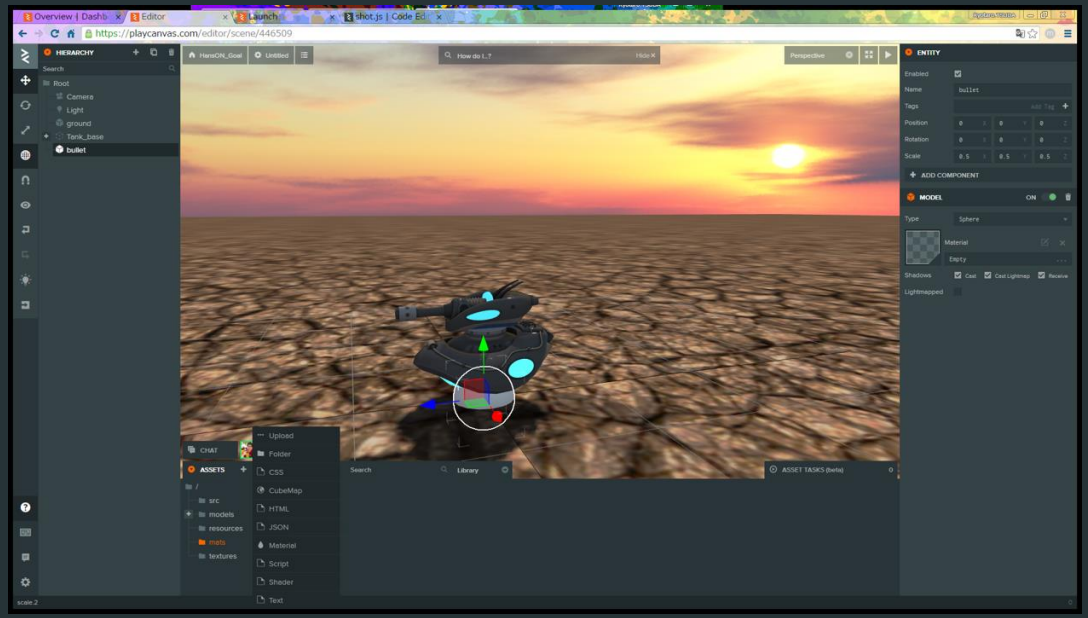
作成後、
INSPECTORから
nameを「bullet」に
Scaleを「0.5, 0.5, 0.5」
に変更します。



12. 弾のEntityを作成する

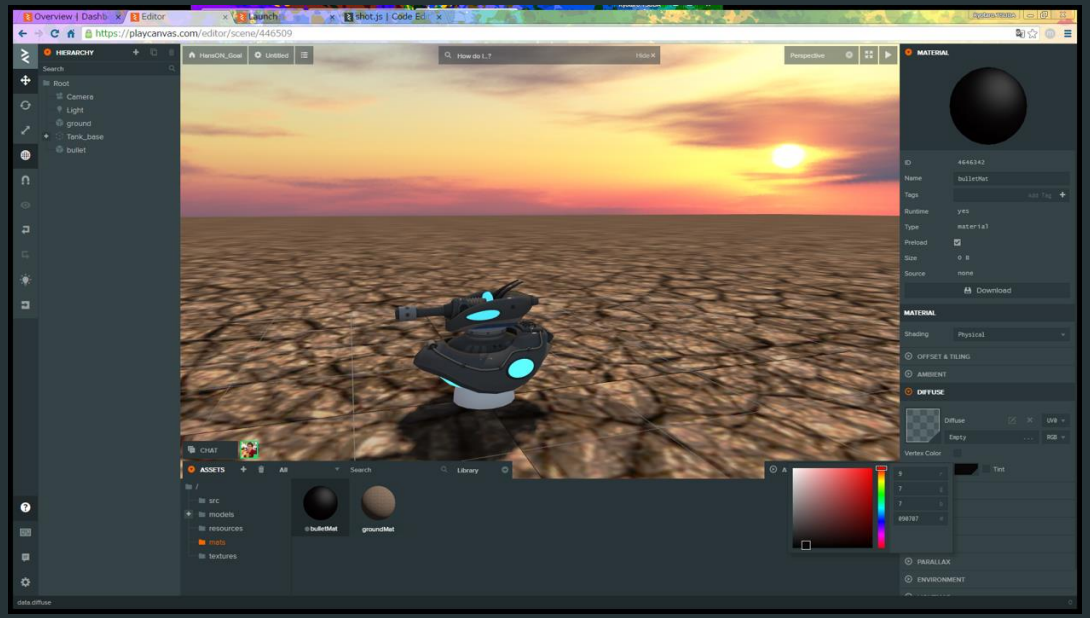
デフォルトだと、色合いが弾っぽくないので、弾のMaterialを作成し適用します。

ASSETSからmatsフォルダーを選択し、＋からMaterialを新規作成し、「bulletMat」と名前をつけます。



13. 弾のEntityを作成する

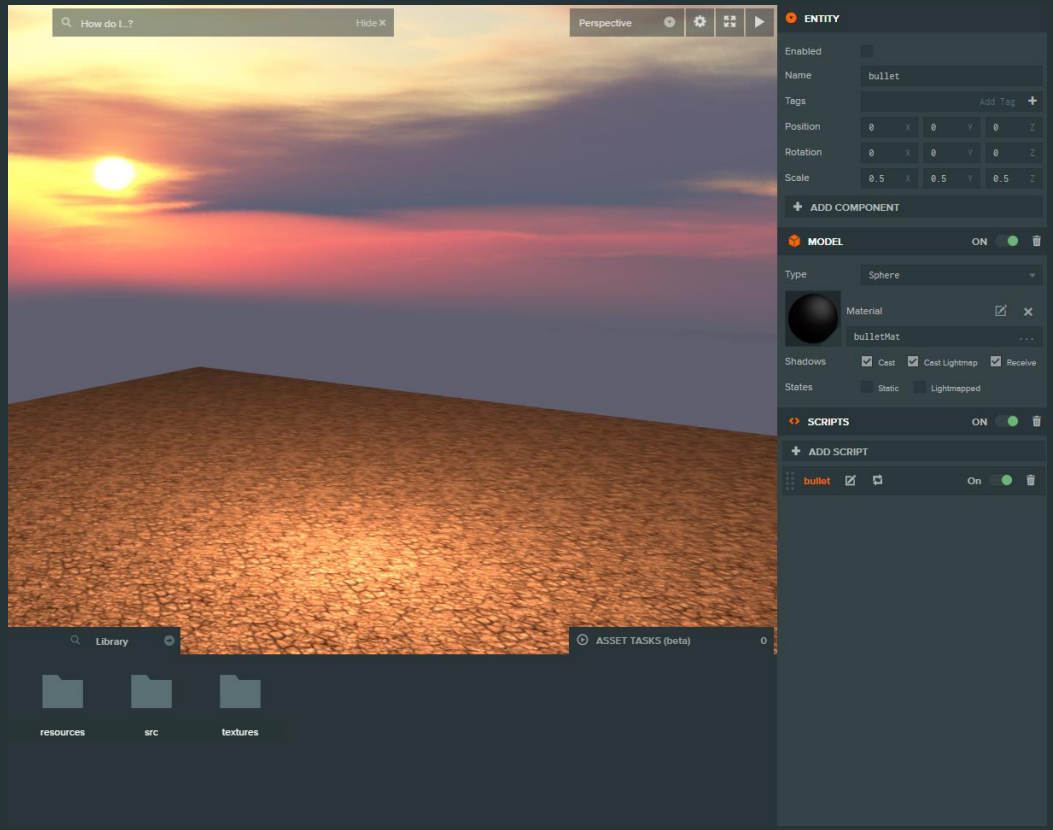
作成後、bulletMatのDIFFUSEからcolorを調整して、黒っぽいMaterialにします。



14. 弾のEntityを作成する

作成後、HIERARCHYからbulletを選択し、INSPECTOR内のADD COMPONETからSCRIPTSを追加します。

「bulletMat」をMODELコンポーネントのMaterialに、「bullet.js」をscriptsコンポーネントにそれぞれアタッチします。



15. 弾のEntityを作成する

最後に弾の挙動を「bullet.js」に記述します。

bullet.jsをCode Editorで開き、右のように記述します。

保存して実行し、弾が飛んで行くことが確認できたら成功です。

bullet.js

```
var Bullet = pc.createScript('bullet');

// initialize code called once per entity
Bullet.prototype.initialize = function() {

};

// update code called every frame
Bullet.prototype.update = function(dt) {
    //z軸方向に1ずつ移動する
    this.entity.translate(0,0,1);
};

// swap method called for script hot-reloading
// inherit your script state here
Bullet.prototype.swap = function(old) {

};

// to learn more about script anatomy, please read:
// http://developer.playcanvas.com/en/
```

16. 戦車から弾を発射する

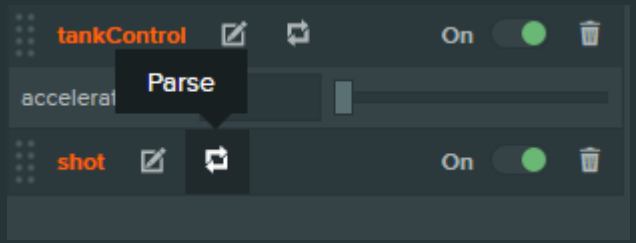
キーボード入力が取得できて、弾のEntityが作成できたので、次はその二つを結び付けます。

playCanvasはPrefabの機能を持っていませんが、"Template"という考え方があるので、それで実装します。

shot.jsを右のように書き換えます。

shot.jsのattribute(属性)に"bulletTemplate"という値をentity型で追加します。

保存後、Game EditorからTank_baseのSCRIPTSコンポーネントのshot.jsのParseボタンを押します。



shot.js

```
var Shot = pc.createScript('shot');
Shot.attributes.add("bulletTemplate",{type:"entity"});

// initialize code called once per entity
Shot.prototype.initialize = function() {

};

// update code called every frame
Shot.prototype.update = function(dt) {
    if(this.app.keyboard.wasPressed(pc.KEY_SPACE)){
        //Spaceキーを押したら
        this.shot();
    }

};

// swap method called for script hot-reloading
// inherit your script state here
Shot.prototype.swap = function(old) {

};

Shot.prototype.shot = function(){
    //弾を発射する
    var bul = this.bulletTemplate.clone();//bulletTemplateのクローンを作成し変数
bulに格納
    var pos = this.entity.getPosition();//タンクの座標を取得
    bul.setName("clone");//bulのNameをcloneに設定
    this.app.root.addChild(bul);//rootの子オブジェクトとして追加
    bul.setLocalPosition(pos.x,pos.y+0.5,pos.z);//bulの位置をタンクの座標に合わせる
    bul.enabled = true;//bulを有効化
};

// to learn more about script anatomy, please read:
// http://developer.playcanvas.com/en/
```

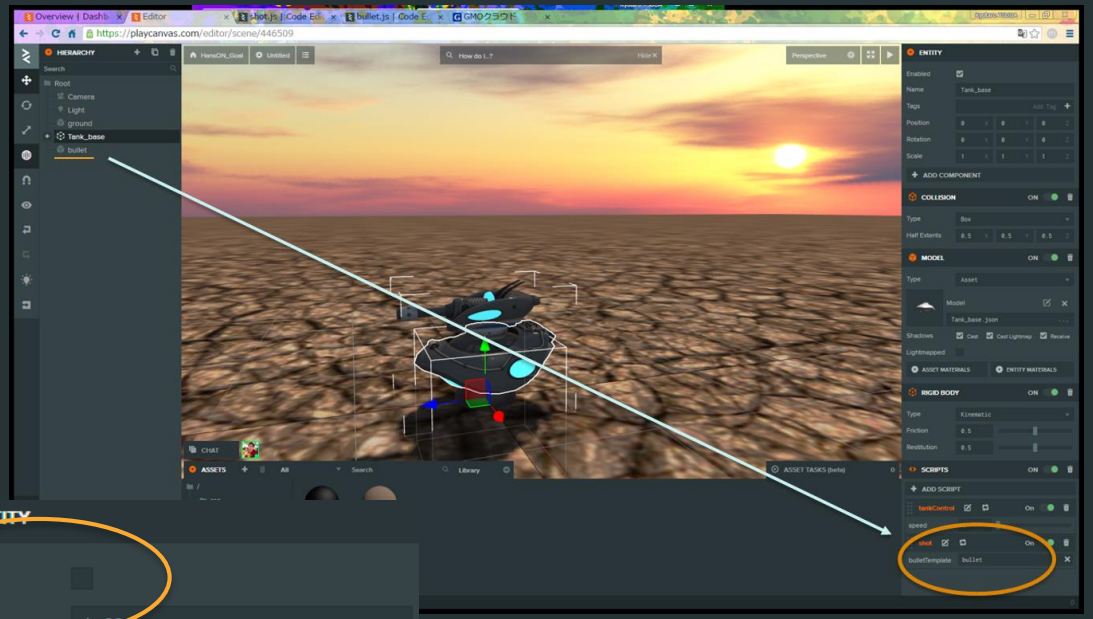
17. 戦車から弾を発射する

するとshot.jsの属性にEntity型でbulletTemplateが追加されます。

HIERARCHYからbulletをbulletTemplateにドラック＆ドロップし、適用させます。

Templateの元になっているbulletは必要ないので、INSPECTORのENTITY内のEnabledのチェックをはずします。

実行後、スペースキーを押すと、戦車の位置から一方向に弾が発射されることが確認できます。



18. 向いている方向に弾を撃つ

戦車の向いている方向にたまを撃つような処理を記述して行きます。

戦車の向いている方向はTank_base(entity).lookAtForにvec3型であらかじめ以下のように格納されています。

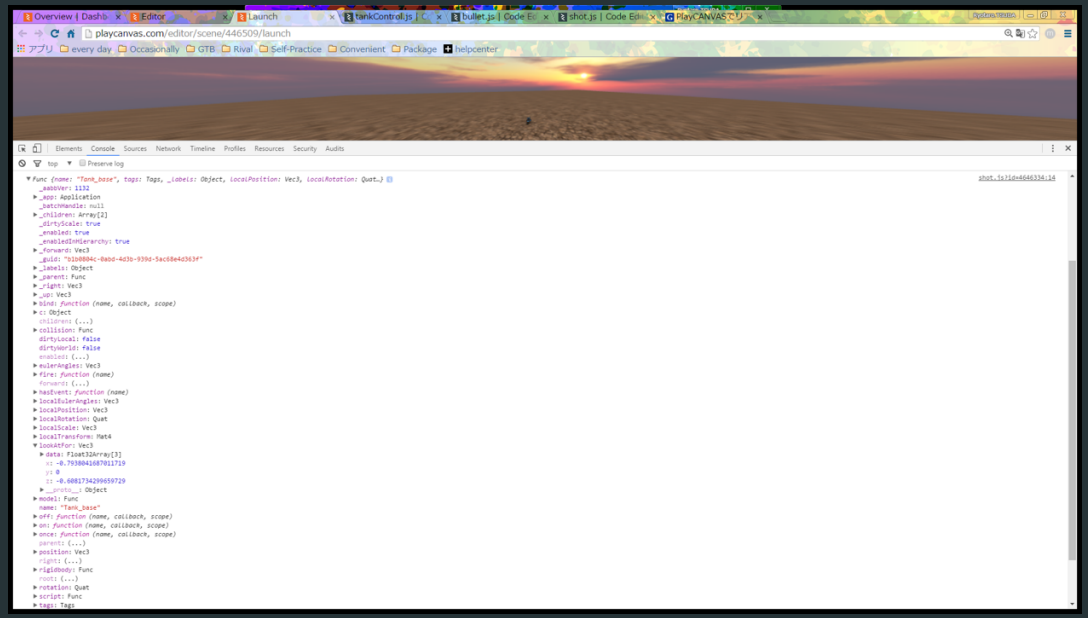
Tank_base

-> lookAtFor

-> x：-0.7938…

-> y：0

-> z：0.6081…..



19. 向いている方向に弾を撃つ

この値を利用して弾の進行方向を変更します。
bullet.jsを開き、右のように書き換えます。

保存して実行すると、向いている方向に弾が発射されることが確認できると思います。

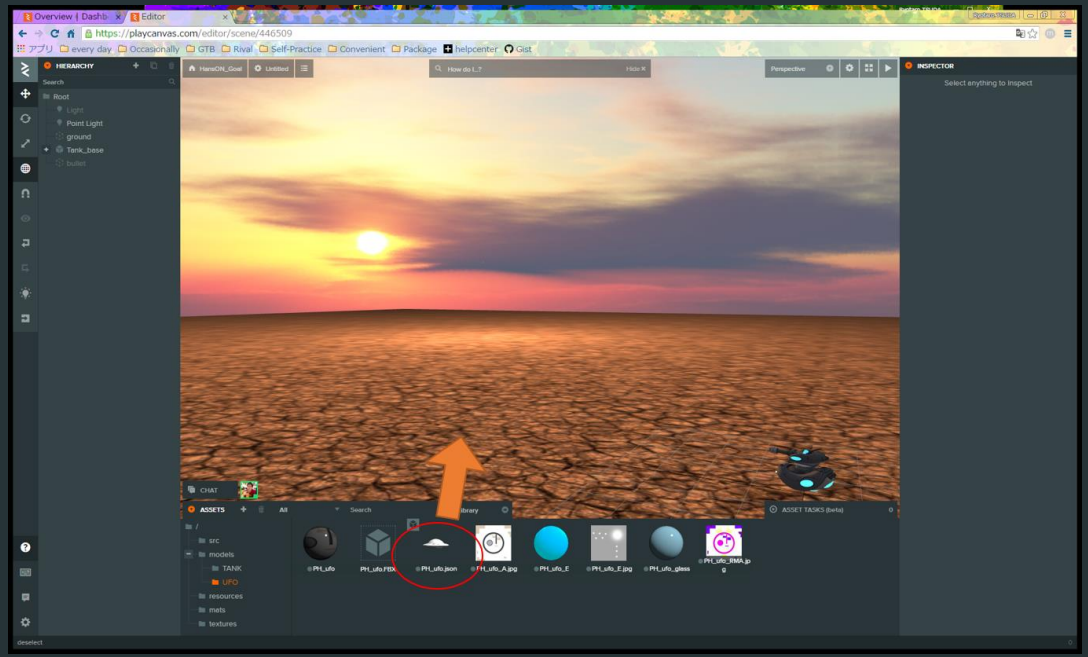
bullet.js

```
var Bullet = pc.createScript('bullet');

// initialize code called once per entity
Bullet.prototype.initialize = function() {
    //Tank_baseを探してlookAtForをthis.lookatforに代入する
    this.lookatfor = this.app.root.findByName("Tank_base").lookAtFor;
};

// update code called every frame
Bullet.prototype.update = function(dt) {
    //向いている方向に飛ばす
    this.entity.translate(this.lookatfor);
};

// swap method called for script hot-reloading
// inherit your script state here
Bullet.prototype.swap = function(old) {

};

// to learn more about script anatomy, please read:
// http://developer.playcanvas.com/en/
```

20. 敵を作る

弾をぶつける敵を作ります。

HIERARCHYからRootを選択して、ASSETS内のmodels -> UFOフォルダ内のPH_ufo.jsonをSCENEにドラックアンドドロップします。



21. 敵を作る

INSPECTORを操作して以下のように設定します。

Entity

Position:[-10, 0, 10]

Scale:[0.5, 0.5, 0.5]

<ADD COMPONENT>

***Collision**

Half Extents:[1, 0.5, 1]

***Rigid Body**

Type:[Kinematic]

***Scripts**

[+ADD Script]

-> +New Script

enemy.js

22. 敵を作る

enemy.jsをCode Editorから開き、右のように記述します

enemy.js

```
var Enemy = pc.createScript('enemy');

// initialize code called once per entity
Enemy.prototype.initialize = function() {

};

// update code called every frame
Enemy.prototype.update = function(dt) {
    //回転する力を加える
    this.entity.rigidbody.angularVelocity = new pc.Vec3(0,50,0);
    this.entity.collision.on("collisionstart", this.death, this);
};

// swap method called for script hot-reloading
// inherit your script state here
Enemy.prototype.swap = function(old) {

};

Enemy.prototype.death = function(result){
    if(result &&
       result.other.rigidbody &&
       result.other.name === "clone"){
          //衝突したコリジョンを持った相手の名前が"clone"だったら
          this.entity.destroy();//自分自身をdestroy
          result.other.destroy();//衝突した相手をdestroy
      }
};


// to learn more about script anatomy, please read:
// http://developer.playcanvas.com/en/
```

23. 敵を作る

最初の状態だと、bulletは剛性を持っていないので、

collisionとrigidbodyを追加しINSPECTORを操作して以下のように設定します。

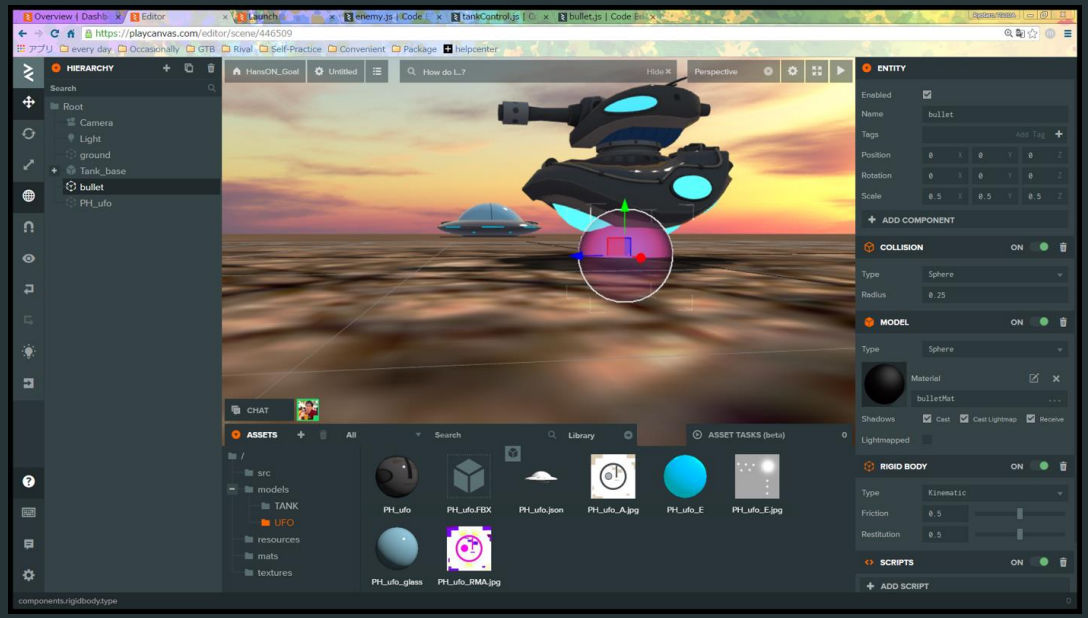
Collision

Type:[Sphere]

Radius:[0.25]

Ridid body

Type:[Kinematic]



24. 敵を作る

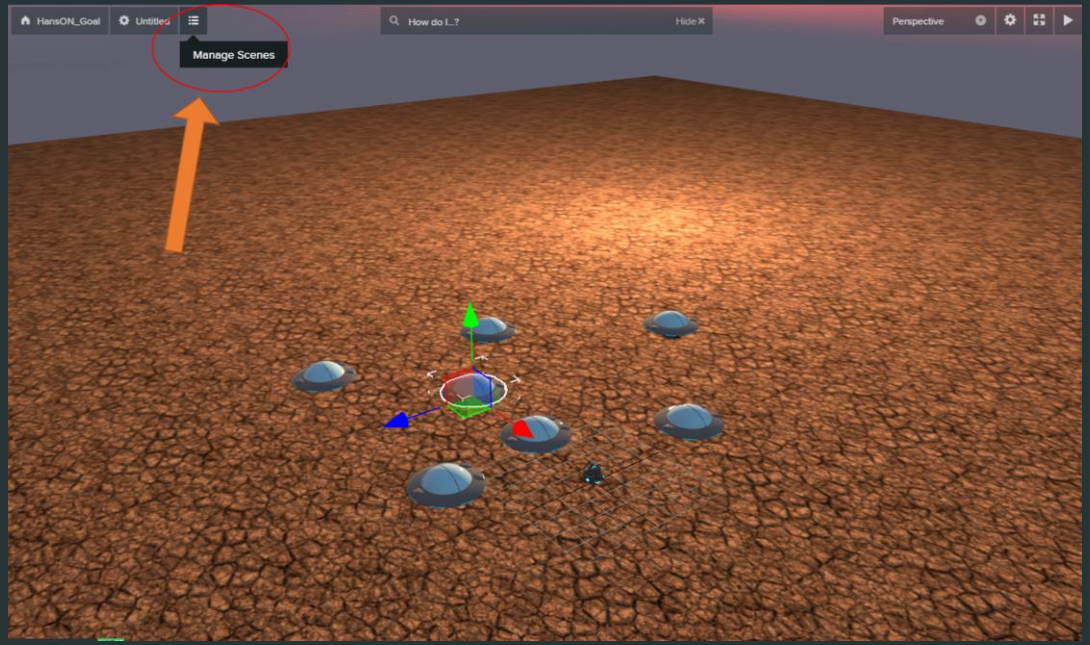
実行して、弾が衝突するとUFOが消えることが確認できたらOKです。

一体じゃつまらないのでHIERARCHYのDuplicate Entityから複製して、たくさん配置してみましょう

25. ゲームを公開する

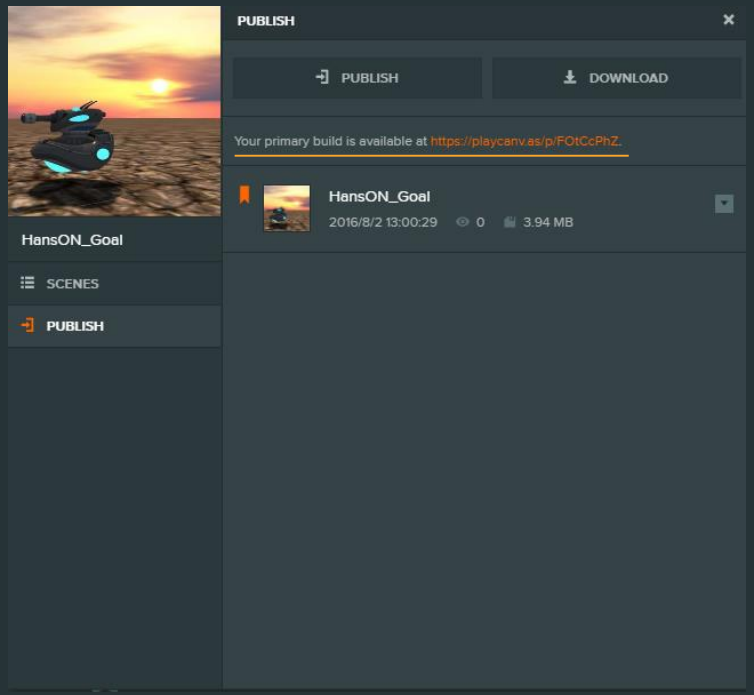
ゲームが完成したら、

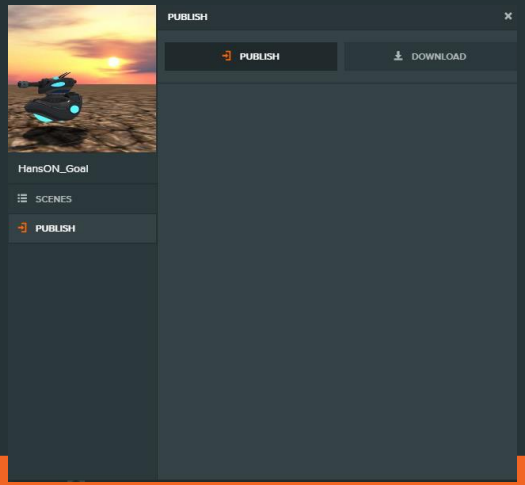
SCENE内のManageScenesから
ゲームをPUBLISHします。

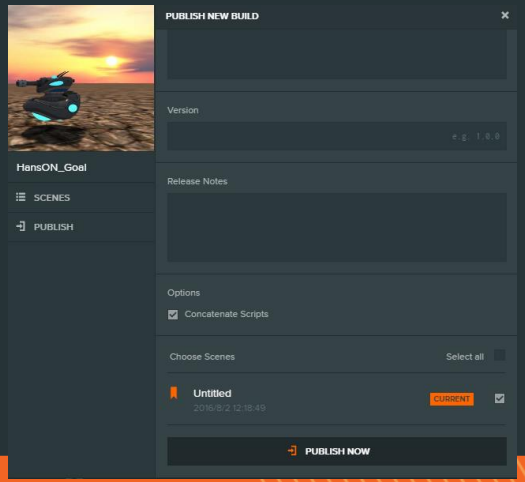


26. ゲームを公開する

PUBLISHタブからPUBLISHを選択します。
諸情報を入力して、PUBLISH NOWを選択すると、URLが発行されます。







ゲームを公開する

- 発行されたURLからゲームを遊ぶことができます。
- たくさんシェアしてたくさん遊んでもらいましょう！