Learnカンペ

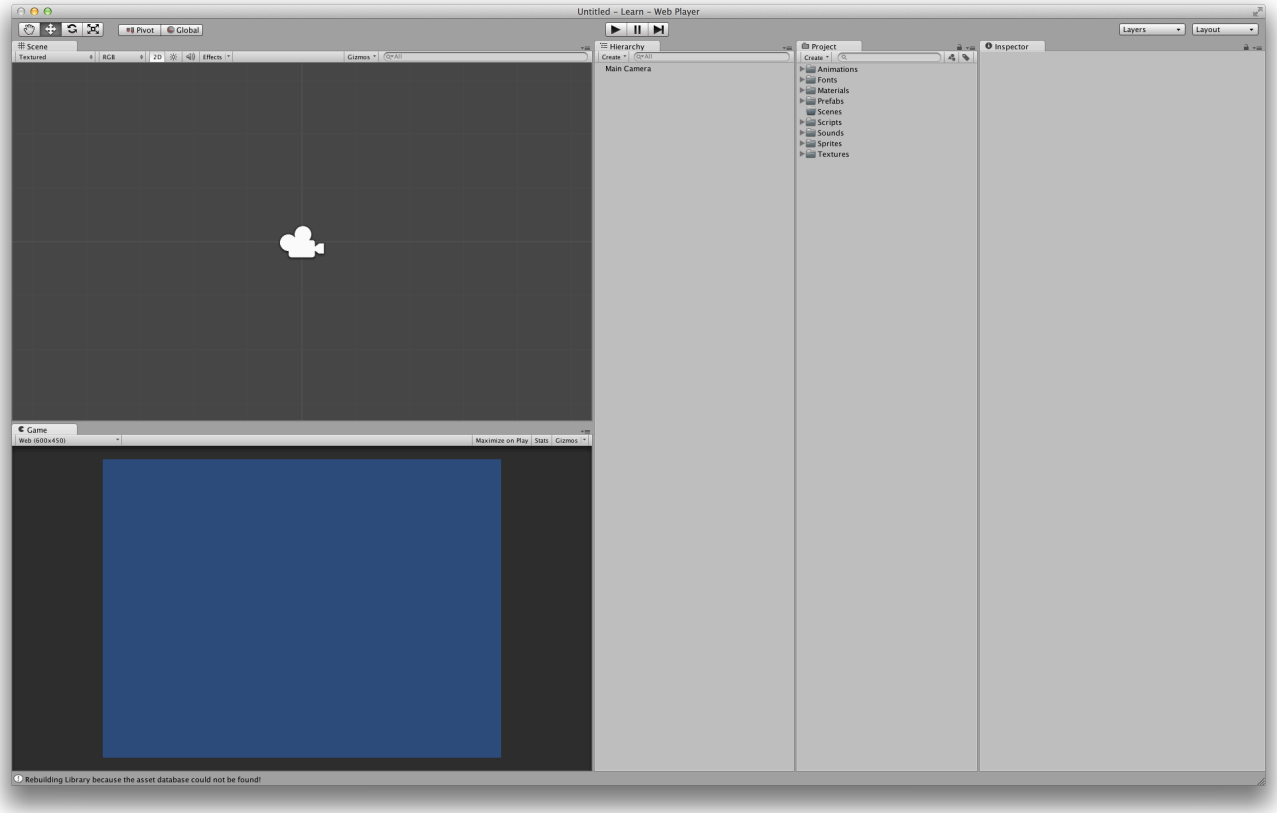
Unityの起動

ShootingGameフォルダの中の**Learnプロジェクト**を開こう。

Unityエディターの設定

手順通りに説明していきます。

手順		
1	レイアウト	2 by 3
2	Projectブラウザ	One Column Layout
3	ビルドターゲット	WebPlayer
4	スクリーンサイズ	600 x 450
5	2Dモード	オン



Main Camera を入れ替える

1. **Prefabs**フォルダに**Main Camera**のプレハブがあります。シーン内にあるMainCameraと入れ替えてください。
    - カメラの環境を同じにするためです。

スプライトの設定 (Sprite Editorを使う)

1. Spaceship 1のTexture Import SettingsのSprite ModeをMultipleに変更します。
2. Texture Import Settingsの「SpriteEditor」ボタンを押してSpriteEditorを開いてみましょう。
3. スプライトを複数のスプライトにわけます。
    - 左上のスライスをクリックしてください
    - 以下の設定でスライスします。

name	value
Type	Grid
Pixel Size	X 48 Y 48

    - Sliceボタンを押してテクスチャに枠線が付いたことを確認しApplyボタンを押して反映します。
4. 同じように `Bullet_16x16` も設定します。
    - Sprite ModeをMultipleにしてSprite Editorを開きましょう。
    - 以下の設定でスプライトをわけます。

name	value
Type	Grid
Pixel Size	X 16 Y 16

5. `Explosion` も設定します。
    - Sprite ModeをMultipleにしてSprite Editorを開きましょう。
    - 以下の設定でスプライトをわけます。

name	value
Type	Grid
Pixel Size	X 64 Y 64

スプライトアニメーション

1. `Spaceship_0` から `Spaceship_3` の `4つ` のスプライトをシーンにドラッグ＆ドロップしてスプライトアニメーションを作成します。
    - `Spaceship_0` を選択してシフトキーを押しながら `Spaceship_3` を選択するとまとめて選択できるよ！
2. アニメーションファイルの名前と保存先を決めるダイアログが表示されます
    - 保存先は `Animations/Player` で、名前を `Normal` として保存しましょう。
3. この時に作成されたゲームオブジェクトの名前を `Player` にします。
    - `Animations/Player/Spaceship_0` も `Player` に変更しましょう。
4. もう既にスプライトアニメーションは完成しています。プレイボタンを押してゲームを再生してみましょう。
5. 同じようにしてエネミーとエクスプロージョンのアニメーションも作成しましょう

スプライト	アニメーション名	アニメーションコントローラー名 ゲームオブジェクト名
Spaceship_0 Spaceship_1 Spaceship_2 Spaceship_3	Normal	Player
Spaceship_4 Spaceship_5 Spaceship_6 Spaceship_7	Normal	Enemy
Explosion_0 Explosion_1 Explosion_2 Explosion_3 Explosion_4 Explosion_5 Explosion_6 Explosion_7 Explosion_8 Explosion_9 Explosion_10	Explode	Explosion

6. 作成したゲームオブジェクト `Player` `Enemy` `Explosion` をプレハブにしましょう。

プレイヤーの移動

1. Playerゲームオブジェクトを矢印キーで操作するために必要なコンポーネントをゲームオブジェクトにつけていきます。

アタッチするもの	説明
Spaceship.cs	スクリプト
Rigidbody2D	Spaceship.cs にRequireComponentで指定している
Player.cs	スクリプト

2. 必要なパラメーターを設定します。

パラメーター名	値
Apply Root Motion	チェック外す
Gravity Scale	0

Monodevelopの起動

1. 機体を移動させるスクリプトを記述します。そのためにスクリプトを書くためのエディタ「Monodevelop」を起動します。
    - Unity上で「Scripts/Spaceship/Spaceship.cs」をダブルクリックしましょう。
        - 開かない場合は「Sync Monodevelop Project」を行い、slnファイルを直接開いてください。
        - Monodevelopの左側にソリューションが開いておらずスクリプトファイルのみの場合はslnファイルから開きなおしてください。

スクリプトの記述

1. 今回はハンズオンのため時間短縮やタイプミスを防ぐためにコピペで済ませます。配布している `スクリプト.html` を開いてください。

    - ShootingGame.zip解答したらルートにファイルがあるはずです。無い時はUSBなどでデータを渡してください。

2. Spaceship.csの「移動するコード」をコピペします。
3. Player.csの「Spaceshipコンポーネントを取得するコード」と「移動と移動範囲を制限するコード」をコピペします。

パラメーターの調整

1. Playerゲームオブジェクトのインスペクター上に表示されているパラーメーター変更していきます。
    - `Spaceship` のSpeedを0から5にしてください

2. ゲームを再生してプレイヤーが矢印キーで動くか確認してみましょう。プレイヤーのスピードは自由に変更して構いません。

プレイヤーの弾を作る

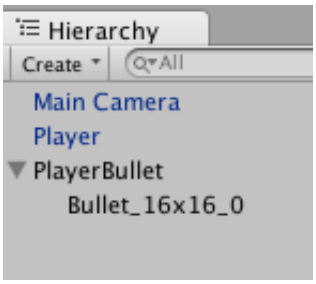
1. [GameObject] -> [Create Empty]からからのゲームオブジェクトを作成し、名前を `PlayerBullet` とします。
2. 必要なコンポーネントを追加します。

アタッチするもの	説明
Bullet.cs	スクリプト
Rigidbody2D	

3. 必要なパラメーターを設定していきます。

パラメーター名	値
Gravity Scale	0

4. プロジェクトの `Bullet_16x16_0` をシーン内にドラッグ＆ドロップして `PlayerBullet` の子要素にします。



5. 名前を `Bullet_16x16_0` から `Bullet` へ変更しましょう。
6. `Bullet` を複製して２つのBulletの位置を `(0.14, 0, 0)` と `(-0.14, 0, 0)` へと変更します。
    - 弾がどの位置にあるかを確認するためにPlayerゲームオブジェクトとPlayerBulletゲームオブジェクトの位置を `(0, 0, 0)` へと移動します。
7. `スクリプト.html` の `Bullet.cs` にある `移動するコード` をコピペします。
8. 必要なパラメーターを設定していきます。

パラメーター名	値
Is Positive	チェック
Speed	10

9. ゲームを再生してみましょう。上に弾が移動しましたか？
10. 最後にプレハブにしてシーン上にあるPlayerBulletゲームオブジェクトを削除します。

プレイヤーから弾を撃つ

1. `スクリプト.html` の `Spaceship.cs` にある `弾を撃つコード` をコピペします。

2. `スクリプト.html` の `Player.cs` にある `弾を撃つコード` をコピペします。
3. 必要なパラメーターを設定していきます。

パラメーター名	値
Bullet	PlayerBulletプレハブ
ShotDelay	0.05

4. ゲームを再生してみましょう。プレイヤーから弾が発射されましたか？

エネミーの移動

1. Enemyプレハブをシーンビューへドラッグ＆ドロップしてエネミーを作成します。
2. 必要なコンポーネントを追加します。

アタッチするもの	説明
Spaceship.cs	スクリプト
Rigidbody2D	Spaceship.cs にRequireComponentで指定している
Enemy.cs	スクリプト

3. `スクリプト.html` の `Enemy.cs` にある `AnimatorとSpaceshipコンポーネントを取得するコード` と `移動するコード` をコピペします。

4. 必要なパラメーターを設定していきます。

パラメーター名	値
Apply Root Motion	チェック外す
Gravity Scale	0
Speed	0.5

エネミーの弾を作る

1. プロジェクトの `Bullet_16x16_1` をシーン内にドラッグ＆ドロップして作成されたゲームオブジェクトの名前を `EnemyBullet` にします。
2. 必要なコンポーネントを追加します。

アタッチするもの	説明
Bullet.cs	スクリプト
Rigidbody2D	

3. 必要なパラメーターを設定していきます。

パラメーター名	値
Gravity Scale	0
Speed	2

4. これでエネミーの弾が出来ました。
5. EnemyBulletゲームオブジェクトをプレハブにしてゲームオブジェクトは削除しましょう。

エネミーから弾を撃つ

1. Enemyゲームオブジェクトのパラメーターを設定します。

パラメーター名	値
Bullet	EnemyBulletプレハブ
Shot Delay	1
Can Shot	チェック

2. 今回はエネミーは様々な角度や個数の弾を撃つようにしましょう。
3. `スクリプト.html` の `Enemy.cs` にある `弾を撃つコード` を追加します。
    - Startメソッドに追加コード
    - Shotメソッドの追加
4. Enemyゲームオブジェクトの子要素として空のゲームオブジェクトを配置します。名前は `ShotPosition` とします。
5. 必要なパラメーターを設定していきます。

パラメーター名	値
Transform(Position)	(0, 0, 0)

6. ゲームを再生してみましょう。弾を撃てましたか？
7. 弾を３方向に撃つようにしてみましょう。

- ShotPositionオブジェクトを２つ分複製しましょう。
- それぞれのパラメータを以下のように修正します。
  - ShotPosition その１

パラメーター名	値
Transform(Rotation)	(0, 0, 20)

  - ShotPosition その２

パラメーター名	値
Transform(Rotation)	(0, 0, -20)

8. 最後にEnemyゲームオブジェクトのインスペクターでApplyボタンを押してプレハブを更新しましょう。

プレイヤーの当たり判定

1. Playerゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	説明
Box Collider 2D	

   - これが当たり判定となるものです。

2. `Box Collider 2D` のパラメーターを変更します。

パラメーター名	値
Is Trigger	チェック
Size X	0.02
Size Y	0.02

   - 1ドットの当たり判定になりました

弾の当たり判定

1. `PlayerBullet` と `EnemyBullet` プレハブをシーン上にドラッグしてゲームオブジェクトを作成します。
2. `EnemyBullet` ゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	説明
Circle Collider 2D	

3. `Circle Collider 2D` のパラメーターを変更します。

パラメーター名	値
Is Trigger	チェック
Radius	0.07

4. 片方の `Bullet` ゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	説明
Polygon Collider 2D	

5. `Polygon Collider 2D` のパラメーターを変更します。

パラメーター名	値
Is Trigger	チェック

   - コライダーの形状を変更します
     - `Sprite Renderer` の▶ボタンを閉じてください。
     - シーンビューに戻り、コライダーの緑の線にカーソルを乗せ、shiftを押しながらドラッグします。
       - つまずいた人はやらなくていいよ！

6. `Bullet` の `Polygon Collider 2D` の文字上で右クリックして「Copy Component」を選択します。

7. もう片方の `Bullet` ゲームオブジェクトに同じ内容の `Polygon Collider 2D` を追加します。

   - Transformコンポーネントの文字上で右クリックし「Paste Component As New」を選択します。
   - 同じ形状の `Polygon Collider 2D` が追加されたことを確認して下さい。

8. 最後に書くゲームオブジェクトのインスペクターの「Apply」ボタンを押してゲームオブジェクトを削除します。

スクリプトから当たり判定を検出する

1. `スクリプト.html` の `Spaceship.cs` にある `爆発するコード` を追加します。
2. `スクリプト.html` の `Player.cs` にある `爆発するコード` を追加します。

3. `スクリプト.html` の `Explosion.cs` にある `爆発後、削除するコード` を追加します。
4. Explosionプレハブに `Explosion.cs` を追加します。
5. `Player` ゲームオブジェクトの `Spaceship` のパラメーターを変更します。

パラメーター名	値
Explosion	Explosionプレハブ

6. 次に弾を特定のゲームオブジェクトにしか当たらないようにするためにレイヤーを設定します。
    - レイヤー名の登録は[Edit]->[Project Settings]->[Tags and Layers]で行います。
        - 今回は既に登録されているはずです。
    - 次に下記のゲームオブジェクトにレイヤーを設定します。

ゲームオブジェクト	レイヤー
Player	Player
Enemy	Enemy
PlayerBullet	PlayerBullet
EnemyBullet	EnemyBullet

    - 最後にプレハブ更新のためにApplyをしましょう
7. ゲームを再生してみましょう。プレイヤーが敵の弾に当たると延々と爆発しますが、当たり判定を検出することが出来ました。

爆発アニメーション

1. 爆発のアニメーションをループから1回のみの再生にします。
    - Explodeアニメーションを選択してインスペクターを表示し、パラメーターを変更します。

パラメーター名	値
Loop Time	チェックを外す

2. ゲームを再生してみましょう。ループしなくなったはずです。ですが今回は爆発が終わってもゲームオブジェクトが残り続けてしまっているので、ゲームオブジェクトの削除を行うようにしましょう。

3. Explosionプレハブをシーン上にドラッグ＆ドロップして `Explosionゲームオブジェクトを選択しながら` 、[Window] -> [Animation]を選択し、

Animationウィンドウを開きます。

4. Explodeの最後のフレームにアニメーションイベントを追加しましょう。
    - アニメーションイベントとはアニメーションの最中に何らかのアクションを行いたい時に使用するもの。
    - 今回は爆発が終わってもゲームオブジェクトが残り続けてしまっているので、ゲームオブジェクトの削除に使用します。
5. 最後のフレームにカーソルを合わせてクリックし、赤い線が出たら左側のアニメーションイベント追加ボタンをクリックします。
    - Edit Animation Eventというウィンドウが表示されるのでFunctionには `Destroy` を選択します。
    - これでアニメーションの再生が終わったらゲームオブジェクトを削除するようになりました。

弾の削除

1. 今のままだと弾のゲームオブジェクトが残り続けてしまうので、弾が削除されるエリアを作成します。
2. `スクリプト.html` の `DestroyArea.cs` にある `弾や敵を削除するコード` を追加します。
3. [GameObject]->[Create Empty]で空のゲームオブジェクトを作成し、名前を `DestroyArea (Bullet)` とします。
4. `DestroyArea (Bullet)` ゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	**説明**
DestroyArea.cs	スクリプト
Box Collider 2D	

5. `DestroyArea (Bullet)` ゲームオブジェクトのパラメーターを変更します。

パラメーター名	**値**
レイヤー名	DestroyArea (Bullet)
Box Collider 2D の Size	X 9 Y 1

6. ゲームを再生してみましょう。ゲーム再生と同時にプレイヤーが爆発してますか？もしくは `DestroyArea (Bullet)` ゲームオブジェクトに突っ込むとプレイヤーが爆発してますか？ですがこのDestroyAreaは弾を削除するためのものなのでプレイヤーを爆発しないようにしましょう。
    - [Edit]->[Project Settings]->[Physics 2D]を選択します。
    - インスペクターの `Layer Collision Matrix` を見てください。これがレイヤーで当

たり判定を制御するものです。
    - `Player` と `DestroyArea (Bullet)` の組み合わせでチェックが付いていると思います。それを外すことによってそのレイヤー同士は当たり判定が発生しなくなります。

7. `DestroyArea (Bullet)` ゲームオブジェクトのパラメーターを変更します。

パラメーター名	値
Box Collider 2D の Center	X 0 Y 3.7

    - これで画面外に配置されるようになりました。

8. ゲームを再生してみましょう。弾が `DestroyArea (Bullet)` ゲームオブジェクトの所で削除されているのがわかります。ですが `PlayerBullet` ゲームオブジェクトが残り続けているので削除しましょう。

    - 方法は様々ありますが今回はスクリプトで5秒後に削除というざっくりとした実装をしてみましょう。
    - `スクリプト.html` の `Bullet.cs` にある `5秒後に破棄するコード` を追加します。

エネミーの当たり判定

1. Enemyゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	説明
Polygon Collider 2D	

2. `Enemy` のパラメーターを変更します。

パラメーター名	値
Is Trigger	チェック
Explosion	Explosionプレハブ

    - コライダーの形状を変更します
        - `Sprite Renderer` の▶ボタンを閉じてください。
        - シーンビューに戻り、コライダーの緑の線にカーソルを乗せ、shiftを押しながらドラッグします。
            - つまずいた人はやらなくていいよ！

3. `スクリプト.html` の `Spaceship.cs` にある `ヒットポイント（HP）を持たせるコード` を追加します。

4. `スクリプト.html` の `Enemy.cs` にある `ダメージ → 爆発するコード` を追加します。
5. HPが0になればエネミーは爆発するようになります。Enemyゲームオブジェクトを選択し、パ

ラメーターを変更します。

パラメーター名	値
Hp	4

6. ゲームを再生してみましょう。敵は倒せましたか？
7. ログとして黄色いワーニングが表示されていると思います。今はそのまま無視してください。

敵をまとめて出現させるエミッターを作る

1. 敵をまとめて１つのWaveとして出現させます。
    - [GameObject]->[Create Empty]で空のゲームオブジェクトを作成し、名前を `Wave` とします。
    - EnemyゲームオブジェクトをWaveゲームオブジェクトの子要素としましょう。
    - これをプレハブにしてWaveゲームオブジェクトを削除しましょう。
2. [GameObject]->[Create Empty]で空のゲームオブジェクトを作成し、名前を `Emitter` とします。
3. Emitterゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	説明
Emitter.cs	スクリプト

4. `スクリプト.html` の `Emitter.cs` にある `敵を出すコード` を追加します。
5. `Emitter` ゲームオブジェクトのパラメーターを変更します。

パラメーター名	値
Transform (Position)	(0, 3.5, 0)
Waves Size	1
Waves Element 0	Waveプレハブ

6. ゲームを再生してみましょう。今回はループものなので敵を倒したら再度敵が出現するようになりましたか？

エネミーの削除

1. 弾の削除と同じように下の画面外に敵が通り過ぎたら削除するようにします。
2. [GameObject]->[Create Empty]で空のゲームオブジェクトを作成し、名前を `DestroyArea (Enemy)` とします。

3. `DestroyArea (Enemy)` ゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	説明
DestroyArea.cs	スクリプト
Box Collider 2D	

4. `DestroyArea (Enemy)` ゲームオブジェクトのパラメーターを変更します。

パラメーター名	値
レイヤー名	DestroyArea (Enemy)
Box Collider 2D の Size	X 9 Y 1
Box Collider 2D の Center	X 0 Y -5

5. ゲームを再生してみましょう。敵が下に行ったとき削除され、上からエネミーが再度出現します。

休憩！

時間がなければココで終わり

または手を止めて聞いてもらうだけにする

ゲーム管理のマネージャーを作る

1. 今回のゲームで行いたいことは
    - 最初はタイトルだけが表示されていること
    - Xキーを押してゲームが開始されること。つまりゲームスタート。

- ゲームスタートと同時にプレイヤーと敵が出現すること
- プレイヤーが死んだらタイトルに戻ること。つまりゲームーオーバー

2. `スクリプト.html` の `Manager.cs` にある `ゲームスタート / ゲームオーバーを管理するコード` を追加します。

3. [GameObject]->[Create Empty]で空のゲームオブジェクトを作成し、名前を `Manager` とします。

4. Managerゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	**説明**
Manager.cs	スクリプト

5. インスペクターで設定すべきものが有りますが、今はこのままで次に進みましょう

タイトルを作成する

1. Xキーを押してゲームを開始するためにタイトルを作成します。
2. [GameObject]->[Create Empty]で空のゲームオブジェクトを作成し、名前を `Title` とします。
3. `スクリプト.html` の `Title.cs` にある `コード` を追加します。

4. Titleゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	**説明**
Title.cs	スクリプト

5. [GameObject]->[Create Empty]で空のゲームオブジェクトを作成し、名前を `GUI` とします。

6. GUIゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	**説明**
GUIText	

7. `GUI` ゲームオブジェクトのパラメーターを変更します。

パラメーター名	**値**
Anchor	upper center
Font	SAM_5C_27TRG_
Font Size	60

8. `GUI` ゲームオブジェクトを `Title` ゲームオブジェクトの子要素とします。そして `GUI` ゲームオブジェクトを複製して2つにしてください。

9. 様々なパラメーターを調整します。

    - `Title` ゲームオブジェクト

パラメーター名	値
Transform (Position)	(0, 0, 0)

    - `GUI` ゲームオブジェクト　（1つめ）

パラメーター名	値
Transform (Position)	(0.5, 0.625, 0)
Text	Shooting Game

    - `GUI` ゲームオブジェクト　（2つめ）

パラメーター名	値
Transform (Position)	(0.5, 0.45, 0)
Text	Press X

10. `Manager` ゲームオブジェクトのパラメーターを変更します。

パラメーター名	値
Title	`Title` ゲームオブジェクト

ゲームとして完成させよう

1. `Player` ゲームオブジェクトを削除します。
2. `スクリプト.html` の `Player.cs` にある `ゲームオーバーのコード` を追加します。
3. `スクリプト.html` の `Emitter.cs` にある `タイトルを表示している間は敵を出さないコード` を追加します。
4. `Manager` ゲームオブジェクトのパラメーターを変更します。

パラメーター名	値
Player Prefab	`Player` プレハブ

5. ゲームを再生してください。とりあえずゲームとしてひと通りの流れができたはずです。

6. プレイヤーの開始位置が中心なので `Player` プレハブの位置を変更します。

パラメーター名	値
Transform (Position)	（0, -2, 0）

音をつける

1. `Sounds/BGM` にあるbgmのアセットをシーンビューへドラッグ＆ドロップしてください。
    - ゲームを再生してみてください。BGMが流れるはずです。
    - ループ再生を行うためにパラメーターを変更します。

パラメーター名	値
Loop	チェック

2. `PlayerBullet` プレハブに `Sounds/SE` にあるshootのアセットをドラッグ＆ドロップしてください。
    - ゲームを再生してみてください。プレイヤーが弾を撃つ時に音が出るはずです。
    - ちょっと音が大きいのでボリュームやピッチを変更します。
    - `PlayerBullet` プレハブを選択してください

パラメーター名	値
Volume	0.1
Pitch	0.64

3. `Explosion` プレハブに `Sounds/SE` にあるboomのアセットをドラッグ＆ドロップしてください。
    - ゲームを再生してみてください。プレイヤーが弾を撃つ時に音が出るはずです。

背景をつける

1. 背景は３枚用意されていて、それぞれスピードの異なる多重スクロールとなっています。
    - 背景をスクロールするには2Dのシステムでは現在行うことが出来ず、3Dの機能を使用します。

2. [GameObject]->[Create Other]->[Quad]でゲームオブジェクトを作成し、名前を `Front 1` とします。
3. [GameObject]->[Create Other]で空のゲームオブジェクトを作成し、名前を `Backgrounds` として、 `Front 1` を子要素としましょう。

4. `Front 1` ゲームオブジェクトにコンポーネントを追加します。

アタッチするもの	説明
Background.cs	スクリプト

5. `スクリプト.html` の `Background.cs` にある `背景を動かすコード` を追加します。

6. `Front 1` ゲームオブジェクトのパラメーターを変更します。

パラメーター名	値
Transform (Position)	(0, 0, 0.1)
Transform (Scale)	(8, 6, 1)
Mesh Renderer の Materials の Element 0	`Background-Front 1` マテリアル

7. ゲームを再生してください。背景が動きましたか？

8. 同じように残り２つの背景を追加します。 `Front 1` を２つ複製してください。それぞれのパラメーターを変更していきます。

    - `Front 2` ゲームオブジェクト

パラメーター名	値
Transform (Position)	(0, 0, 0.2)
Mesh Renderer の Materials の Element 0	`Background-Front 2` マテリアル
Speed	0.08

    - `Back` ゲームオブジェクト

パラメーター名	値
Transform (Position)	(0, 0, 0.3)
Mesh Renderer の Materials の Element 0	`Background-Back` マテリアル
Speed	0.02

休憩！

時間がなければココで終わり
または手を止めて聞いてもらうだけにする

敵がダメージを受けた時に色を変える

1. 敵がダメージを受けた時に、わかりやすいように機体の色を変更します。
2. `Enemy` プレハブをシーンビューにドラッグ＆ドロップしてゲームオブジェクトを作成します。
3. `Enemy` ゲームオブジェクトを選択しながら、[Window]->[Animation]を選択してアニメーションビューを表示します。
4. `Damage` アニメーションを作成します。
    - アニメーションビューで「Create New Clip」を選択します。
    - `Animations/Enemy` に名前を `Damage` として保存します。
    - `Damage` アニメーションのLoop Timeはチェックを外しましょう。
5. `Enemy` ゲームオブジェクトを選択しながら、[Window]->[Animator]を選択してアニメータービューを表示します。
6. 左下の「Parameters」の `+` ボタンをクリックし、 `Trigger` を選択します。
    - トリガー名を `Damage` としましょう。
7. Normalステート上で右クリックし、 `Make Transition` を選択します。そしてDmageステート上までカーソルを移動しクリックしてください。
8. 作成されたTransitionの矢印をクリックしてください。そしてインスペクターの `Conditions` を `Exit Time` から `Damage` へ変更します。
9. Damageステート上で右クリックして `Make Transition` を選択し、Normalステートへつなげてください。ContitionsはExit Timeのままで問題ありません。
10. この段階で正しい動きをしているか確認します。 `Enemy` ゲームオブジェクトのHPを1000にして死ににくくしてください。そしてゲームを再生します。 `Enemy` ゲームオブジェクトを選択したままにしてください。

11. ステートが遷移していることがわかると思います。次にDamageアニメーションをダメージを受けた時に色を変更するようにします。
12. `Enemy` ゲームオブジェクトを選択しながら、Damageアニメーションが選択されていることを確認し、Add Curveボタンを押してください。

    - Sprite RendererのColorを選択します。

13. **1:00**の所にあるキーフレームを削除して、 **0:00**にあるキーフレームを **0:10**へドラッグして移動させます。

14. キーフレームの色情報を変更します。

パラメーター名	値
r	1
g	0
b	0.6
a	1

15. この状態でゲームを開始してみましょう。弾を当てた時、敵の色は変化しましたか？

16. 最後に `Enemy` ゲームオブジェクトを削除しましょう。

プレイヤーを無敵状態でスタートさせる

1. ２秒間の間無敵状態でスタートさせます。
2. `Player` プレハブをシーンビューにドラッグ＆ドロップしてゲームオブジェクトを作成します。
3. アニメーションビューで「Create New Clip」を選択し、 `Animations/Player` に名前を `Invincible` として保存します。

    - `Invincible` アニメーションのLoop Timeはチェックを外しましょう。

4. Invincibleアニメーションが選択されていることを確認し、Add Curveボタンを押してください。

    - Sprite RendererのEnabledを選択します。
    - Enabledのチェックを**2:00**まで交互にチェックを付けるようにしましょう。

        - キーフレームはコピペできるので楽！

    - 次にAdd Curveボタンを押し、Box Collider 2DのEnabledを選択します。
        - **0:00**はチェックを外し、**2:00**にチェックを入れます。

5. アニメータービューでInvincibleステート上で右クリックし、 `Set As Default` を選択しま

す。

6. Invincibleステート上で右クリックし、 `Make Transition` を選択します。そしてNormalステート上までカーソルを移動しクリックしてください。

7. 試しに `Wave` の `Enemy` のHPを1000、Delay Shotを0.01にして確かめてみましょう。

8. 確認できたら `Enemy` の値を戻しておきましょう。

弾幕にする

1. [GameObject]->[Create Empty]で空のゲームオブジェクトを作成し、名前を `Rotate` とします。
2. `スクリプト.html` の `Rotate.cs` にある `回転するコード` を追加します。
3. `Rotate` を `Enemy` の子要素、 `Shotposition` を `Rotate` の子要素とします。
4. `Rotate` と `Shotposition` の位置は（0,0,0）となるようにしましょう。
5. これでもいいのですがもう少し弾を増やしましょう。
6. `Shotposition` を３つから８つに増やします。
7. `Shotposition` のZ軸を４５度ずつずらしてください。

n個目	値
1	0
2	45
3	90
4	135
5	180
6	225
7	270
8	315

8. `Enemy` のShot Delayを `0.2` にします。

9. ゲームを再生してみてください。

終