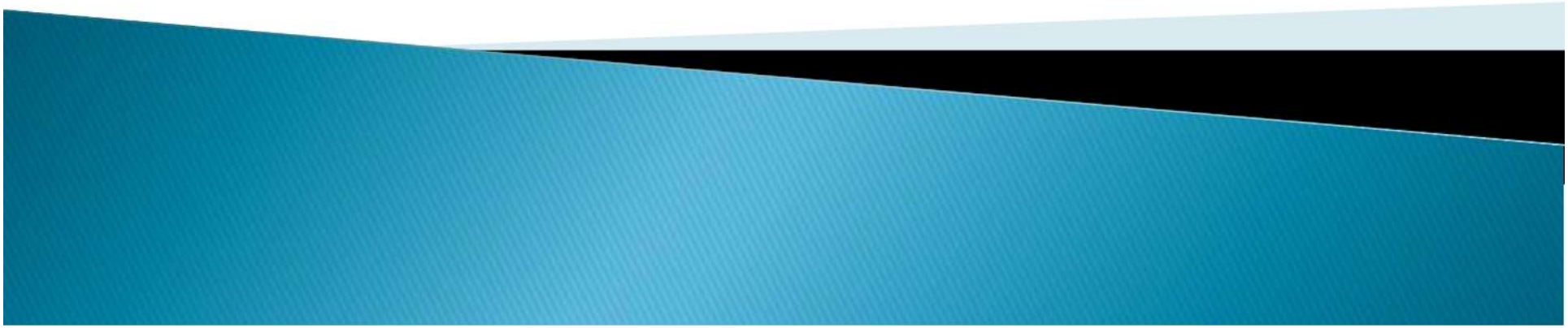
C#でゲームを作ろう2017

第8回　担当:ten



自己紹介

- ID:ten
- 京都大学工学部情報学科2回生
  - 計算機科学コース
- 第40代会長
- ゲーム制作して競プロしてる
- パズルと音ゲー

Twitter

Slackとか

ゲームを考えてきてね

- 夏休み後あたりに発表会をする予定
- 夏休み前に内容を終えるので、そこから自分のゲームを作ってもらう予定
- というわけでゲームの内容を考えてきてください
- いわゆる「３分ゲー」のようなもので大丈夫です

Slack

- #csgame

Github

- https://github.com/kmc-jp/csgame2017

今日やること

- シューティングの続きをつくる

- 前回までで作ったもの
  - プレイヤーが動く
  - プレイヤーが弾を撃つ
  - 敵が動く

今日やること

- 今回やること
  - 敵の弾を作る
  - シーンやレイヤを分ける
  - 当たり判定を実装する
  - テキストを表示する
- 次回までかかるかも
  - というか次回までやる予定

今日やること

- 今回登場する概念
  - C#のはなし
    - null
    - var
    - is演算子
    - as演算子
    - this
  - Altseedのはなし
    - シーン
    - レイヤ
    - Color
    - フォントジェネレータ
    - 動的にフォントを作る

敵弾を作る

- BulletをコピーしてEnemyBulletにする

シーンとレイヤのおはなし

- https://github.com/altseed/STGLecture/blob/master/Document/cs/11.md
- ここの図で

GameSceneを作る

- ゲームそのもののシーン
- プレイヤーと敵のレイヤ
- 弾のレイヤ

TitleSceneを作る

- テキスト表示
- シーン遷移

テキストについて

- TextObject2D
  - (TextureObject2Dを変換しようとしたらでてくるアレ)

- フォントジェネレータの使い方
- 動的にフォントを生成

フォントジェネレータ

- Altseed_CS_110_WIN¥Toolに入っている

フォントジェネレータ

- テキストファイル
  - そのフォントで表示する文字を列挙したtxtファイル
  - Text.txt
- 出力先ディレクトリ
  - .affと.pngを出力するディレクトリの選択
  - Fontsみたいなディレクトリをどこかに作るとよい？
- シート名
  - 出力したファイルの名前がこれになる

- これ以外は大体見た目通りの挙動をします

注意事項

- フォントにはライセンスがある
- 配布するゲームに使えない場合がある

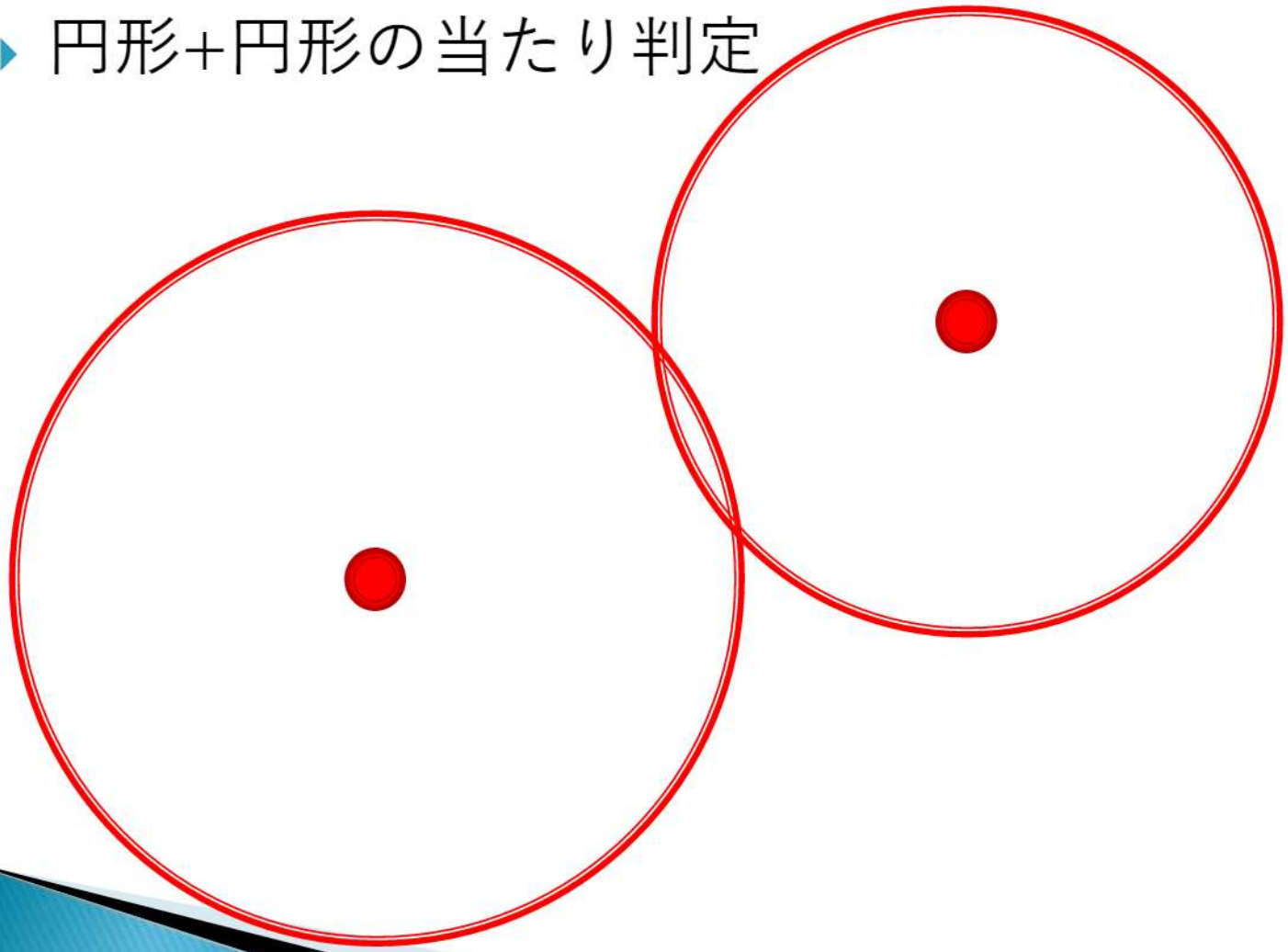
動的にフォント生成

- コードの中でフォントを作ることができる
- この場合任意の文字が使える（はず）
- フォント名を""(空欄)にするとOSのデフォルトのフォントが使える
- PCに依存するので、リリース時は使えなさそう？

```
text.Font = Engine.Graphics.CreateDynamicFont("MS Gothic", 24,
    new asd.Color(0, 0, 0, 255), 2, new asd.Color(255, 255, 255, 255));
```

当たり判定のおはなし

- 円形+円形の当たり判定



当たり判定の実装

- CollidableObjectの実装
  - TextureObject2Dを継承
  - Player,Enemy,Bullet,EnemyBulletはCollidableObjectを継承

当たり判定の実装

- EnemyとBulletの当たり判定
  - 主にEnemyでの実装

```
//何かとぶつかったかの判定(追加)
protected void CollideWith(CollidableObject obj)
{
    if(obj is null)
    {
        return;
    }

    if(obj is Bullet)
    {
        if (IsCollide(obj))
        {
            OnCollide(obj);
            obj.OnCollide(this);
        }
    }
}
```

当たり判定の実装

- EnemyとBulletの当たり判定
  - 主にEnemyでの実装
  - （これはOnUpdate内）

```
//弾のレイヤにあるもの全てと当たり判定(追加)
foreach (var obj in gameScene.bulletLayer.Objects)
{
    CollideWith(obj as CollidableObject);
}
```

当たり判定の実装

- これは何をやってるのか

- ゲームシーンの弾のレイヤの中を全部見る
  - var
- それら全てに対しCollideWithを実行
  - CollideObjectを引数にとる関数
  - そのためのas

var

- 暗黙の型
- コンパイラが型を決める
- https://docs.microsoft.com/ja-jp/dotnet/csharp/language-reference/keywords/var

- 今回「gameScene.bulletLayer.Objects」の中には「Object2D」が入っているが、それを知らなくても実装できる
- 他にも使い道がある（コンパイラしか型名知らないやつとか）

当たり判定の実装

- CollideWithの実行

- obj is null
  - Objがnullだった場合何も実行しない
  - さっきのasの処理と関連
  - 関数内でreturn;があると、そこで実行終了

- obj is Bullet
  - objがBulletに型変換できる場合のみ実行

as

- 「obj as CollidableObject」の場合
- objをCollidableObjectとして扱う

- (CollidableObject)obj、でいいのでは？
  - 型変換できなかったときの処理が違う
    - asの場合：null
    - キャストの場合：例外を投げる

当たり判定の実装

- objがBulletだったら当たってるか判定

当たった後のこと

- 弾：消える
- 敵：点滅
  - Colorをいじる

当たった後のこと

- PlayerとEnemyBulletに対しても同じことをする

HPの実装

- HPをもつオブジェクトのクラスを作る
- HPを表示するクラスを作る

ゲームの終了の実装

- ゲームクリアシーン
- ゲームオーバーシーン

お疲れさまでした

- 次回は7/1(土)