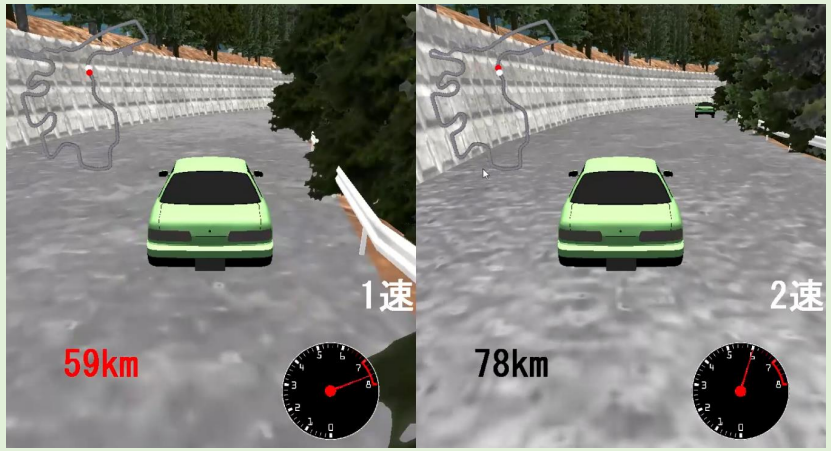
PORTFOLIO

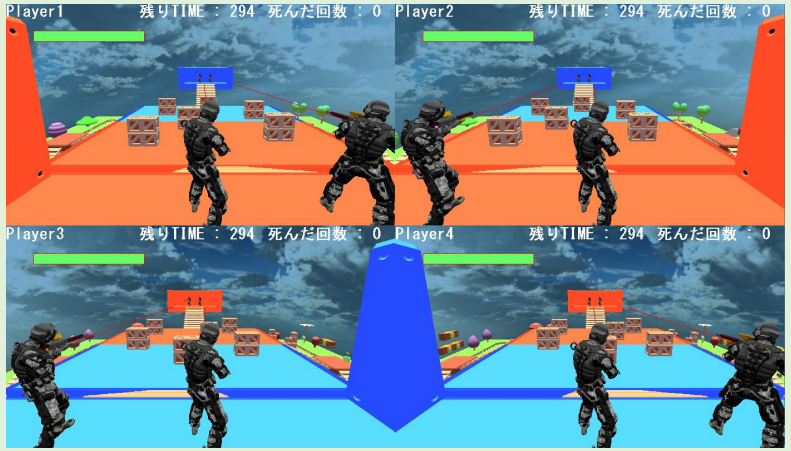
ASOポップカルチャー専門学校

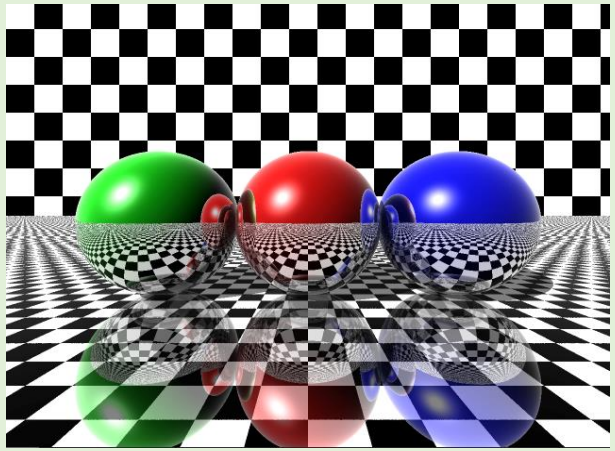
ゲーム･CG･アニメ専攻科 ゲーム専攻

時枝　龍太

TOKIEDA RYUTA







Profile

時枝　龍太

ASOポップカルチャー専門学校

ゲーム・CG・アニメ専攻科　ゲーム専攻

2025年3月卒業見込み

メールアドレス　：　2116016@s.asojuku.ac.jp

自己紹介

私はゲーム制作をする上でコミュニケーションを大切にしてきました。
私にとってコミュニケーションは情報のやり取り以上の意味を持ち、異なる人々との対話から新しいアイデアやひらめきを得てきました。それらのひらめきやアイデアを活かし、楽しいゲームを制作してきました。
主にDXライブラリを用いてC++で開発をしてきました。

Exciteレース

学内コンテスト　二位受賞

学内コンテストに出すために制作した作品です。言語理解を深めるためにエンジンを使わず制作しました。

1～2人プレイです。
1人：タイムアタック　　2人：VSレース

ブレーキ、シフトチェンジをうまく使い競い合うゲームです
ゲームセンターにある頭文字Dアーケードや湾岸ミッドナイトアーケードを参考にゲームを制作いたしました。

Gitにコードをアップロードしています。
https://github.com/tokimarusuisann/ExciteRace.git

制作時期：三年前期

開発環境：DXライブラリ

使用言語：C++

制作期間：三ヶ月

ジャンル　：レースゲーム

カメラ制御

普通の追従カメラだと走っている感じがせず疾走感が全く感じられなかったのでクォータニオンのSlerp(球面線形補間)を使いカメラ制御をしました。

```
199  void Camera::SetBeforeDrawFollow(void)
200  {
201
202      //指定したオブジェクトのポジション
203      VECTOR targetPos = target_->pos_;
204
205      //指定したオブジェクトの回転
206      Quaternion targetRot = target_->quaRot_;
207
208      //今のカメラの状態
209      Quaternion nowCamera = Quaternion::Euler(0.0f, angleY_, 0.0f);
210
211      //今の状態のカメラからプレイヤーの回転まで球面補間
212      slerpRot_ = Quaternion::Slerp(nowCamera, targetRot, SLERP_TIME);
213
214      //カメラ位置と注視点を移動
215      VECTOR relativeCameraPos = slerpRot_.PosAxis(SLERP_RELATIVE_CAMERA_POS);
216      pos_ = VAdd(targetPos, relativeCameraPos);
217      VECTOR relativeTargetPos = slerpRot_.PosAxis(SLERP_RELATIVE_TARGET_POS);
218      targetPos_ = VAdd(pos_, relativeTargetPos);
219
220  }
```

通常追従カメラ

https://youtu.be/wVaLajl5mxU

再生できます

球面線形補間を使った追従カメラ

https://youtu.be/cEsm7d3B2jA

再生できます

衝突判定

車のスピードが速くなると、カプセルとモデルの衝突判定や円とモデルの衝突判定ではすり抜けが発生してしまいました。そのため、車の移動先の座標を用いて線分で衝突判定を行うことで、すり抜けを回避しました。

```
525  void Car::Collision(void)
526  {
527
528      //前方向
529      VECTOR forward = transformCar_.GetForward();
530
531      //線形補間で押し戻す
532      collisionPow_ = AsoUtility::Lerp(collisionPow_, AsoUtility::VECTOR_ZERO, COLLISION_POWER);
533
534      //移動先の座標
535      movedPos_ = VAdd(transformCar_.pos_, VScale(forward, speed_));
536
537      //押し戻した座標を足す
538      movedPos_ = VAdd(movedPos_, collisionPow_);
539
540      //移動前座標と移動先座標の線分で衝突判定を行う
541      for (auto collider : colliders_)
542      {
543          auto hit = MV1CollCheck_Line(collider, -1, transformCar_.pos_, movedPos_);
544
545          //衝突していたら
546          if (hit.HitFlag > 0)
547          {
548              auto& nomal = hit.Normal;
549
550              //高さは無視する
551              nomal = VNorm(nomal);
552              nomal.y = 0.0f;
553
554              //押し戻し処理
555              movedPos_ = VAdd(hit.HitPosition, VScale(nomal, MOVED_POS_SIZE));
556
557              //減速処理
558              Deceleration(SPEED_DOWN_BREAK_POWER);
559          }
560      }
561
562      //更新
563      transformCar_.pos_ = movedPos_;
564
565  }
```

モデルと図形での衝突判定

https://youtu.be/trjMWBY3Ois

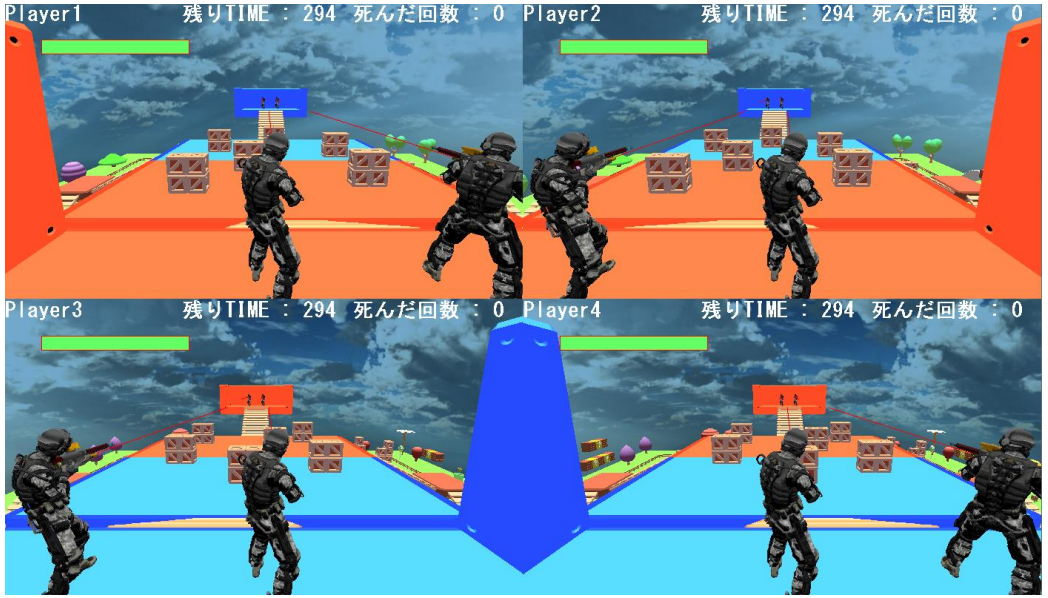
再生できます

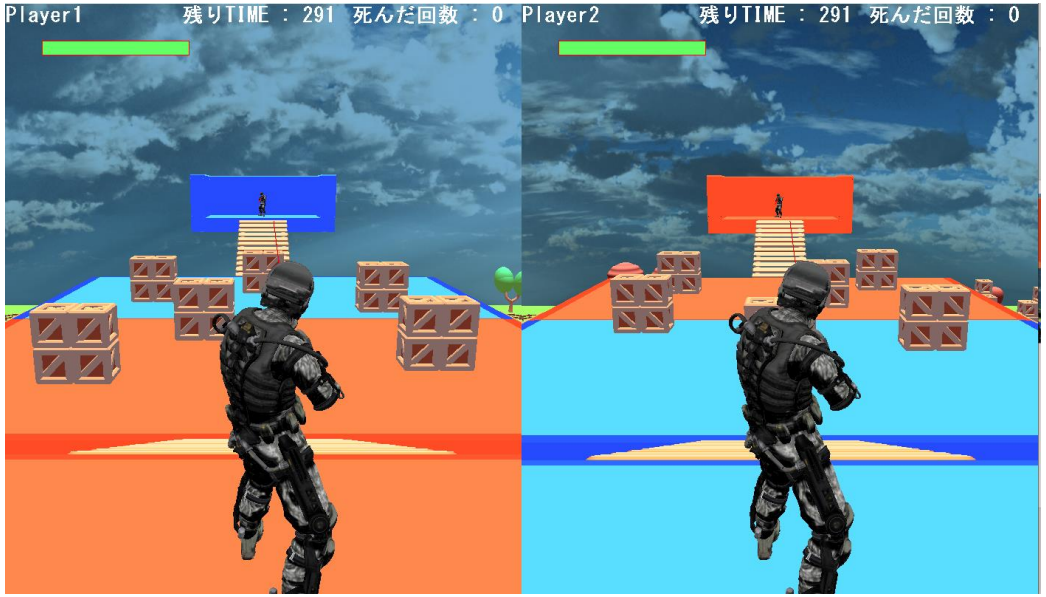
線分での衝突判定

https://youtu.be/B6m0bO1z7z4

再生できます

TPSゲーム





みんなで楽しく遊べるゲームを作りたいという思いで
制作いたしました。

2～4人プレイです。
2人プレイ、3人プレイは個人戦です。
4人プレイはチーム戦です。

画面分割をし1画面で多人数遊べるようにしました。
ボーン制御にこだわった作品です。

制作時期：二年後期

開発環境：DXライブラリ

使用言語：C++

制作期間：三ヶ月

ジャンル　：TPS

このゲームを作る際、ステージに高低差があるため、プレイヤーが上下に球を打ち分けることができるようにしました、その際アニメーションを使わず実際にボーン（フレーム）を動かしました。

クォータニオンによるボーン制御

ボーンを指定して回転させる処理

```
MATRIX Player::MoveFrameRotateMatrix(int mHandle, int moveFrameIndex, float movePow, VECTOR rotDir)
{
    //行列を初期化（前回の行列が残ったままになりおかしくなる）
    MV1ResetFrameUserLocalMatrix(mHandle, moveFrameIndex);

    //ワールド回転をローカル回転に変換するために作った逆回転関数を呼び出し
    Quaternion localReverseRot = GetRotReverseBornWorldToLoacl(mHandle, moveFrameIndex);

    //現在のローカル座標からワールド座標に変換する行列を得る
    auto worldMatrix = MV1GetFrameLocalWorldMatrix(mHandle, moveFrameIndex);

    //上で取ってきた情報に（大きさ、位置、回転が入っている）一応回転成分のみ抽出
    auto worldRotMatrix = MGetRotElem(worldMatrix);

    //背骨のワールドクォータニオンをゲット
    auto worldRotQuaternion = Quaternion::GetRotation(worldRotMatrix);

    //回転させたい回転量
    auto rotPower = Quaternion::AngleAxis(movePow, rotDir);

    //回転させたい回転量に背骨のワールドクォータニオンを合成
    auto mixSpineWorldQuaternion = rotPower.Mult(worldRotQuaternion);

    //逆回転を合成
    auto mixSpineLocalQuaternion = localReverseRot.Mult(mixSpineWorldQuaternion);

    //初期状態の変換行列
    auto initMat = MV1GetFrameBaseLocalMatrix(mHandle, moveFrameIndex);

    //初期状態の拡大率の取得
    auto vecScl = MGetSize(initMat);

    //初期上の移動行列取得
    auto vecPos = MGetTranslateElem(initMat);

    //初期状態の拡大行列の取得
    auto matScl = MGetScale(vecScl);

    //平行移動行列の取得
    auto matPos = MGetTranslate(vecPos);

    //クォータニオンから行列へ変換
    auto multMatrix = mixSpineLocalQuaternion.ToMatrix();

    //単位行列に取得してきた行列を合成
    MATRIX matrix = MGetIdent();
    matrix = MMult(matrix, matScl);
    matrix = MMult(matrix, multMatrix);
    matrix = MMult(matrix, matPos);

    return matrix;
}
```

ねじれをクォータニオンで表すために、親ボーンの座標軸の方向を子ボーンの座標軸に合わせる計算（ワールド空間での回転クォータニオンをローカル空間に変換）

```
Quaternion Player::GetRotReverseBornWorldToLoacl(int mHandle, int moveFrameIndex)
{
    //素のクォータニオン（最終return値）
    Quaternion result = Quaternion::Identity();

    //親フレーム番号
    auto frameParentNum = MV1GetFrameParent(mHandle, moveFrameIndex);

    //親フレームが存在する間（parentFrameが-2ではない）
    while (frameParentNum >= 0)
    {
        //ここでワールド回転をローカル回転に変換 親の行列（大きさ、回転、位置）
        auto localParentMatrix = MV1GetFrameLocalMatrix(mHandle, frameParentNum);

        //親の行列をクォータニオンに変換（回転）
        auto localQuaternionRotParent = Quaternion::GetRotation(localParentMatrix);

        //親の逆回転情報（Inverse()）を合成してあげる、
        result = result.Mult(localQuaternionRotParent.Inverse());

        //ここで計算したフレームに親フレームがあれば回る
        frameParentNum = MV1GetFrameParent(mHandle, frameParentNum);
    }

    //親の逆回転行列を持つresultにプレイヤーの逆回転を合成
    result = result.Mult(transform_.quaRot_.Inverse());
    result = result.Mult(transform_.quaRotLocal_.Inverse());

    return result;
}
```

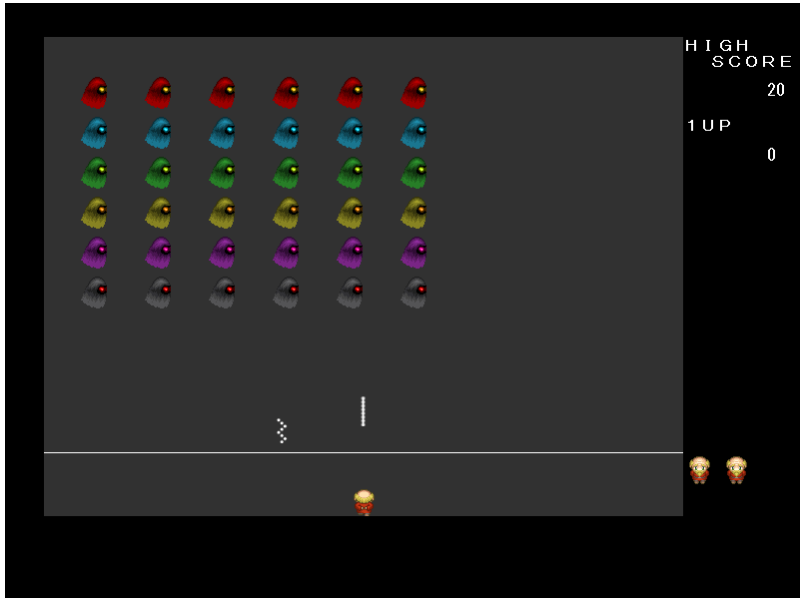
個人＆授業作品紹介

一年次

シューティング

制作時期　　:一年前期
使用言語　　:C言語
使用ライブラリ　:DXライブラリ

初めてのゲームプログラミングでシューティングゲームを制作いたしました。C言語の基礎や、画像の描画、衝突判定、キー入力、ゲームループなどの基本的なことを学びました。背景のスクロールは画像を2枚並べて交互に来るようにして実装しています。

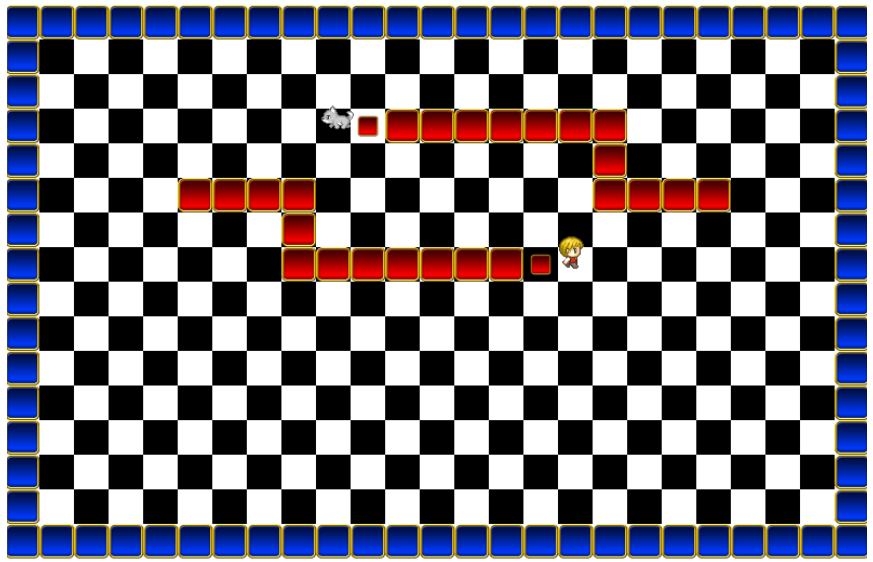


インベーダー

制作時期　　:一年前期
使用言語　　:C言語
使用ライブラリ　:DXライブラリ

インベーダーゲームを制作しました。文字の描画や、スコア、残機、キャラクターのアニメーション、複数の敵を配列で管理する方法などを学びました。ゲームに面白みを持たせるにサウンド、ステージ選択、ボス戦などを実装しました。

一年次



スネークゲーム

制作時期　　:一年後期
使用言語　　:C++
使用ライブラリ　　:DXライブラリ

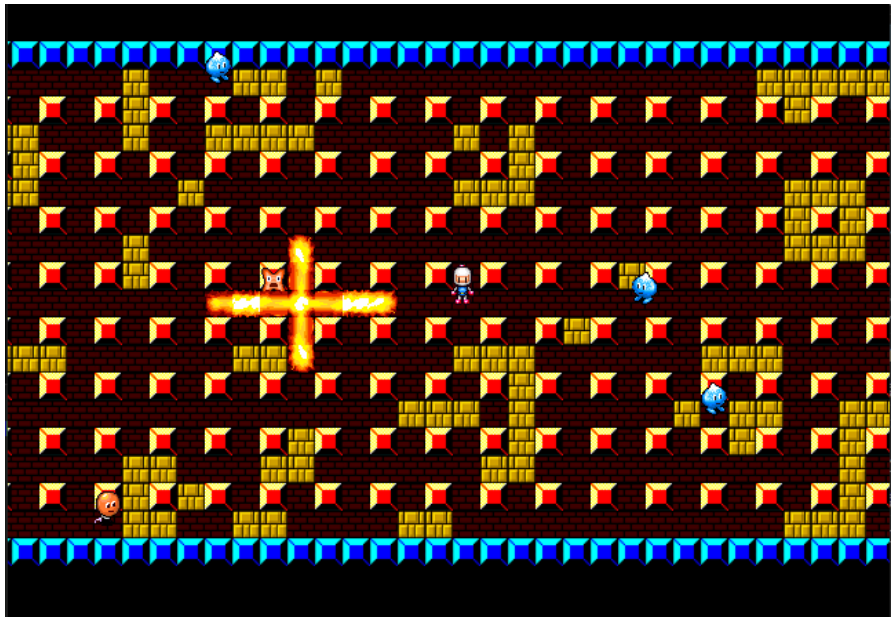
初めてC++言語で制作しました。カプセル化やポリモー フィズム、関数オブジェクト、シングルトン、標準テンプレートライブラリなどを学びました。当たり判定をする為にbool型の二次元配列を用意し、 座標を今いるマス目に計算して勝敗判定を実装しました。

2Dアクション

制作時期　　:一年後期
使用言語　　:C#
使用エンジン　　:Unity

初めてのUnity 制作で2Dアクションを作りました。
カメラの追従、敵や星のジェムをプレハブ化しコピー生成、アニメーターでモーション切り替え、敵を踏んだ時の当たり判定など実装しました。

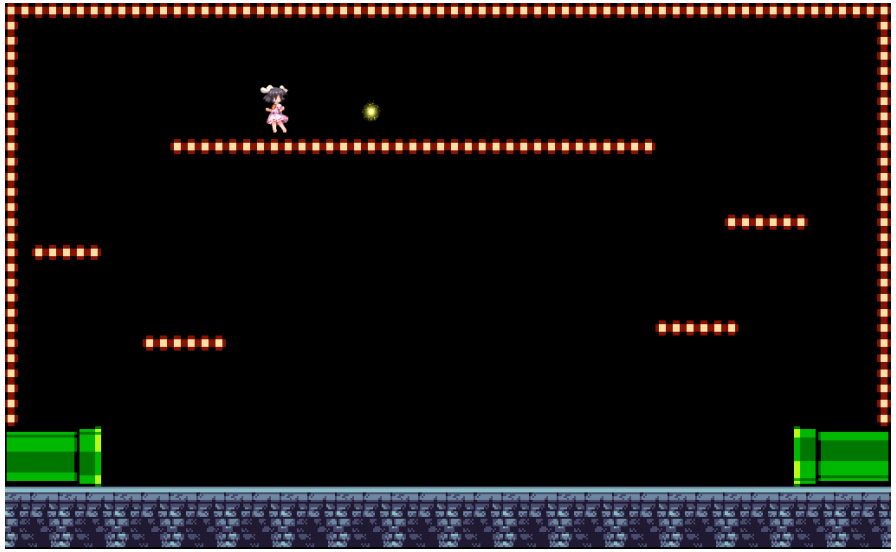
二年次



ボンバーマン

制作時期　　:二年前期
使用言語　　:C++言語
使用ライブラリ　:DXライブラリ

ステージをTiledMapEditorで制作しXML形式で読み込み実装しました。十字に伸びる 爆風や誘爆の実装方法、ラムダ式や継承、STL、関数ポインタ などの使い方を学びました。



コマンドアクション

制作時期　　:二年前期
使用言語　　:C++言語
使用ライブラリ　:DXライブラリ

TiledMapEditorの機能を用いて制作した衝突判定用のオブジェクト レイヤーを読み込み、そのデータを用いてプレイヤーとステージの 衝突処理を実装しました。リングバッファを用いることでコマンド入力 を実装し、波動拳や昇竜拳を撃つことができます。

二年次

学内コンテスト上位入賞

ESCAPEゲーム

制作時期　　:二年前期
使用言語　　:C#
使用エンジン　:Unity

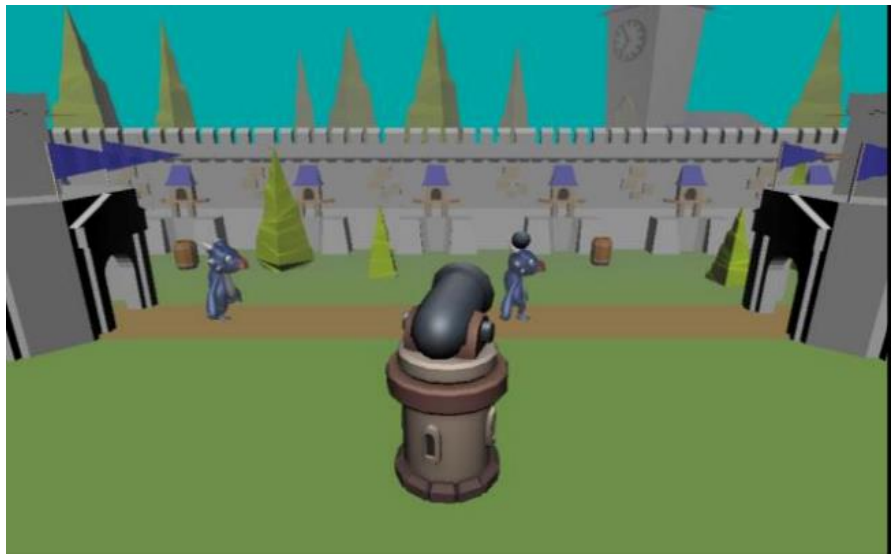
Unityで3Dアクションを作りました。
3Dゲームとしては初めての制作でしたが、C#の関数を用いてプレイヤー追跡、弾発射、アニメーション切り替え等をプログラムで実装しました。

3Dシューティングゲーム

制作時期　　:二年前期
使用言語　　:C++
使用ライブラリ　: DXライブラリ

C++での初めての3Dゲーム制作でした。
クォータニオンを使用した回転、バネの力を応用した時間差でカメラの追従を連動させる処理、ボス戦の前のイベントシーン等を実装しました。

二年次



3Dタワーディフェンス

制作時期　　:二年前期
使用言語　　:C++
使用ライブラリ　:DXライブラリ

3Dゲームの復習として、タワーディフェンスゲームを制作しました。 砲台と砲身の親子関係を設定したり、砲身の角度に合わせて 弾を発射する処理、楽しめるようにステージや敵を複数実装しました。

3Dアクションゲーム

制作時期　　:二年後期
使用言語　　:C++
使用ライブラリ　:DXライブラリ

ステージによって重力方向が変化したり傾斜の角度によって坂道を滑り落ちる処理、また、ステージ間の移動には線形補間を使いマリオギャラクシーのようなステージ移動を実装いたしました。

数学制作作品

数学作品

数学　シューティング

制作時期	:二年前期
使用言語	:C++
使用ライブラリ	:DXライブラリ

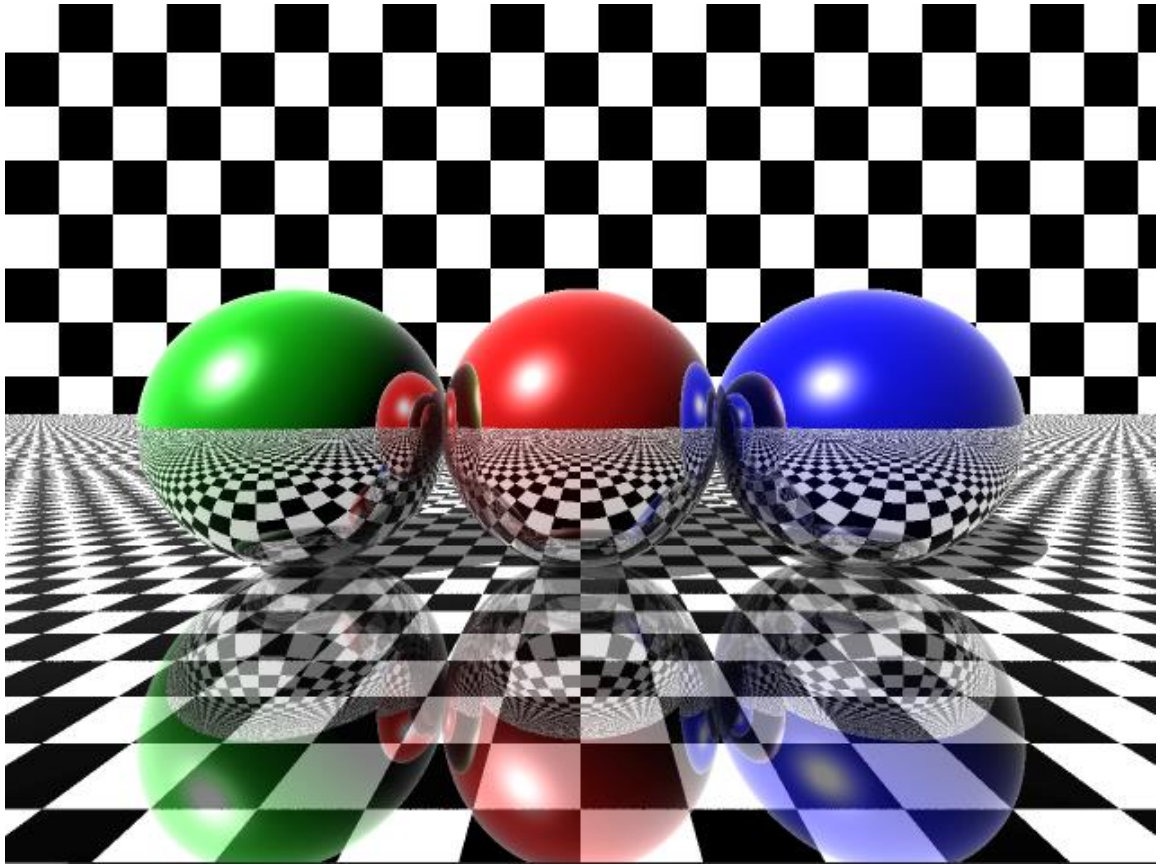
ベクトルから弾幕の発射角度を求め、様々な種類の弾幕を発射できるようにしました。 自機狙い弾、全方位弾、ばら撒き弾等を実装しました。

数学　キャリーログ

制作時期	:二年後期
使用言語	:C++
使用ライブラリ	:DXライブラリ

内積と外積、行列を使った回転を学び、2Dのカプセルの当たり判定を学びました。 ゲームとして遊べるように丸太ゲームを製作しました。 当たり判定を可視化することで遊びやすくしました。

レイトレーシング



制作時期	:二年後期
使用言語	:C++
使用ライブラリ	:DXライブラリ

DXライブラリのピクセルごとに色を指定して描画する機能を使用し、古典的レイトレーシングを製作しました。スクリーンの座標からRayを飛ばして、球体、平面との当たり判定をして、最初に当たったオブジェクトの色情報を取得し1 ピクセルづつ描画しました。 影、鏡面反射、拡散反射などの表現を実装しました

Thank you!