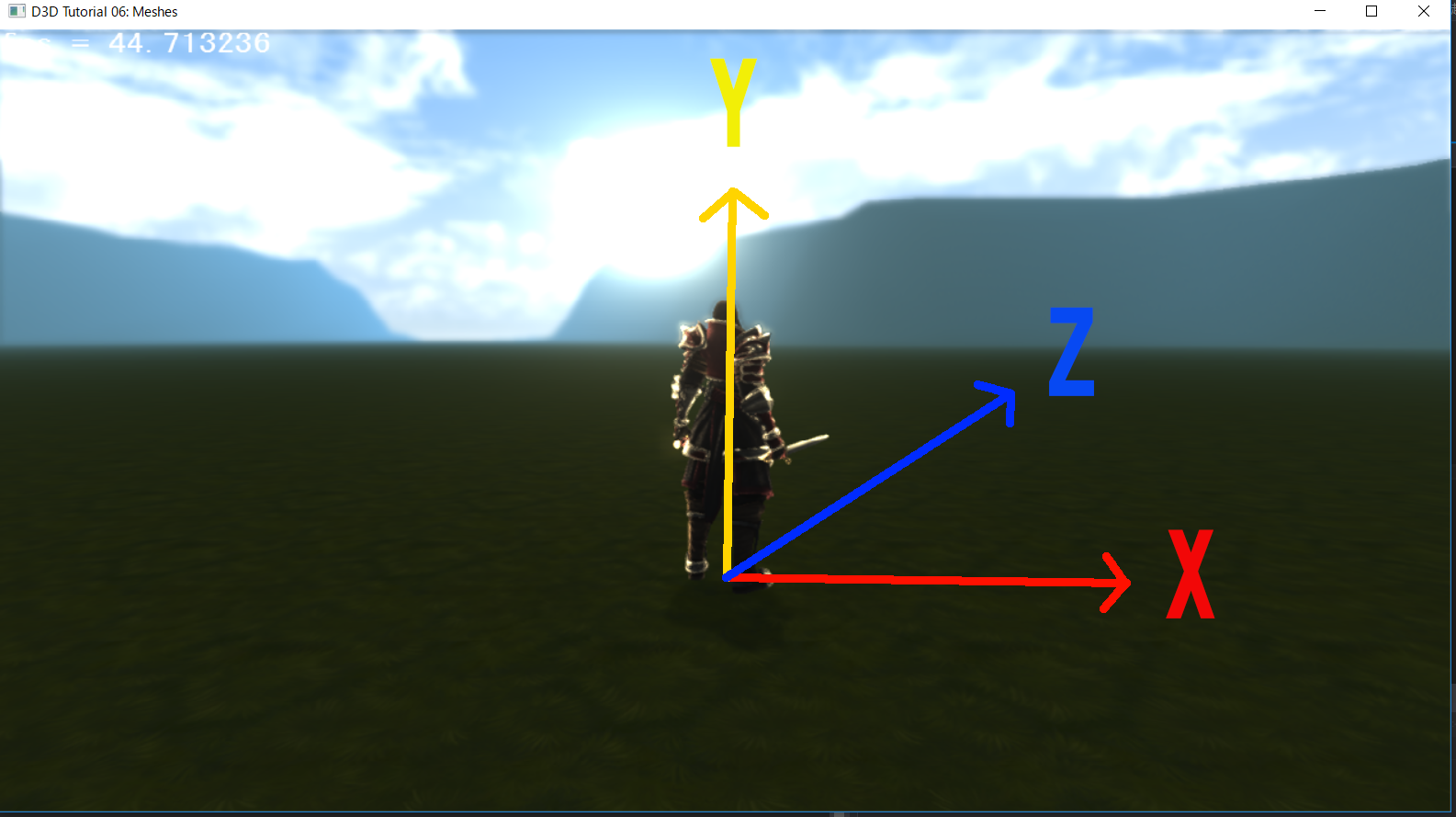
**Lesson\_01\_1 キャラクターを動かしてみよう。**

　3Dゲームでキャラクターの位置は下記の図のように３の軸上の位置で表されます。



この絵のキャラクターはX軸上に0.0、Y軸上に0.0、Z軸上に0.0の場所にいます。では、このキャラクターを少しずつX軸上で動かすプログラムを書いてみましょう。

サンプルプログラムに次のコードを記述してみてください。

Lesson01/Question/Player.cpp

|  |
| --- |
| void Player::Move()  {  //Question 1 キャラを左右に動かしてみよう。  moveSpeed.x **= 0.2f;**  //Question 2 キャラを上下に動かしてみよう。  //Question 3 キャラをジャンプさせてみよう。  //Question 4 重力を加えてみよう。  //移動。  characterController.Execute(moveSpeed, GameTime().GetFrameDeltaTime());  position = characterController.GetPosition();  } |

moveSpeedはキャラクターの移動速度を表す変数です。X軸方向に速度を設定すると、キャラクターはX軸上を移動します。

実習課題(１０分)

「Question 2 キャラを上下に動かしてみよう。」と記述されている次の行にキャラクターをZ軸上で動かすプログラムを改造しなさい。

**Lesson\_01\_2 パッドの入力でキャラを動かしてみよう。**

Lesson\_01\_1の実装ではキャラクターは常に移動を行っていました。多くのゲームではゲームパッドでの入力によってキャラクターが移動すると思います。このチャプターではゲームパッドの入力によって、キャラクターを動かせるようにしてみましょう。

**2.1 if文**

パッドの入力があった場合だけキャラクターを動かしたい場合はif文を使用します。「Question 1キャラを左右に動かしてみよう。」と記述されている箇所に次のコードを記述してみてください。

Lesson01/Question/Player.cpp

|  |
| --- |
| //Question 1 キャラを左右に動かしてみよう。  if (Pad(0).IsPress(enButtonRight)) { //もしもゲームパッドの右ボタンが押されていたら。  moveSpeed.x = 1.0f; //移動速度を１にする。  }else{ //それ以外。  moveSpeed.x = 0.0f; //何も入力されていないので移動速度を０にする。  } |

これでパッドの右ボタンが入力されるとキャラクターが右に移動するようになりました。では、続いて左ボタンが入力されたらキャラクターが左に移動するようにしてみましょう。

Lesson01/Question/Player.cpp

|  |
| --- |
| //Question 1 キャラを左右に動かしてみよう。  if (Pad(0).IsPress(enButtonRight)) { //もしもゲームパッドの右ボタンが押されていたら  moveSpeed.x = 1.0f;　　　　　//移動速度を1にする。  }  else if (Pad(0).IsPress(enButtonLeft)) { //右のボタンが押されていなくて、左のボタンが押されているなら。  moveSpeed.x = -1.0f; //移動速度を-1にする  }  else {  moveSpeed.x = 0.0f; //何も入力されていないので移動速度を０にする。  } |

実習課題(１５分)

Question 2 キャラを上下に動かしてみよう。」と記述されている次の行からプログラムを追加して、パッドの上下のボタンを入力することによってキャラクターがZ軸上を動くようにしなさい。

**Lesson\_01\_3 キャラクターをジャンプさせてみよう。**

　パッドのAボタンが押されたら、キャラクターをジャンプさせてみましょう。ジャンプするということはキャラクターの上方向(y方向)に力が加わるということです。つまり、moveSpeed.yに速度を設定してやればいいことになります。次のコードをサンプルプログラムに記述してみてください。

Lesson01/Question/ Player.cpp

|  |
| --- |
| if (Pad(0).IsTrigger(enButtonA)) { //もしもAボタンが押されたら  moveSpeed.y = 2.0; //上方向に力を加える。  characterController.Jump();//おまじない。気にしないでOK。  } |

どうでしょうか、ジャンプできたのではないでしょうか。しかしキャラクターはジャンプしたまま大気圏を突破したのではないかと思います。

**Lesson\_01\_4 キャラクターに重力を加えてみよう。**

　私たち人間がジャンプをしても大気圏を突破せずに、地面に立つことができるのは地球の引力によって引っ張られているためです。この地球が引っ張る力のことを重力といいます。Lesson\_01\_3でキャラクターが大気圏を突破してしまったのは、重力が加えられていなかったためです。「Question 4 重力を加えてみよう。」と記述されている箇所に、次のプログラムを記述してみてください。

Lesson01/Question/ Player.cpp

|  |
| --- |
| //Question 4 重力を加えてみよう。  moveSpeed.y -= 0.1f; |

どうでしょうか？これでキャラクターは正しく、地面に落下してきたと思います。

**Lesson\_02\_1 キャラクターを回転させてみよう。**

　Lesson\_01ではパッドの入力でキャラクターを移動させました。Lesson\_02ではパッドの入力でキャラクターを回転させてみましょう。

1. **クォータニオン(四元数)**

3Dモデルの回転を表現するにはいくつか手法があるのですが、今回は3Dゲームで主流となっているクォータニオンを使用した回転の表現について見ていきましょう。この授業は数学の授業ではないのでクォータニオンの数学的な定義や証明は行いません。クォータニオンをゲームでどのように使用するのかという点に注視して説明を行います。

回転の表現にクォータニオンを使用する理由の大きな理由の一つに任意の軸周りの回転を簡単に扱うことができるというものがあります。では任意の軸周りの回転とはどのようなものなのか見ていきましょう。例えば、下の図のようにキャラクターを回転させたい場合は、Y軸周りに90度回転させることになります。

では次はX軸周りに回転させてみましょう。

では実際にプログラムを記述して、キャラクターを回転させてみましょう。下記のコードをサンプルプログラムに記述してください。

Lesson\_02/Question/Player.cpp

|  |
| --- |
| void Player::Rotation()  {  //Question 1 キャラクタを右に向かせてみよう。  　rotation.SetRotationDeg(CVector3::AxisY, 90.0f);  //Question 2 キャラクタを左に向かせてみよう。  //Question 3 キャラクタを奥に向かせてみよう。  //Question 4 キャラクタを手前に向かせてみよう。  } |

キャラクターが右に90度回転したと思います。

**Lesson\_02\_2 パッドの入力でキャラクターを回転させてみよう。**

　では、続いてパッドの右ボタンが押されたらキャラクターが右を向くようにしてみましょう。Lesson\_01\_2を思い出してください。パッドの右ボタンが押されたら？という処理を書きたい場合はif文を使うのでしたね？では、「Question 1キャラクターを右に向かせてみよう。」と記述されている箇所に次のコードを記述してみてください。

Lesson\_02/Question/Player.cpp

|  |
| --- |
| if (Pad(0).IsPress(enButtonRight)) { //もしもゲームパッドの右ボタンが押されていたら  rotation.SetRotationDeg(CVector3::AxisY, 90.0f); //90度回す。  } |

どうでしょう、正しく回転できたでしょうか？

実習課題(２０分)

パッドの上下左右のボタンを押すことによって、キャラクターが移動方向を向くようにプログラムを改造しなさい。

**Lesson\_03\_01 走るアニメーションを再生してみよう。**

　Lesson\_02までの実装でキャラクターが移動して、進行方向を向いて回転するようになりました。しかし、キャラクターが立ちのままで移動しているため不自然です。ではプログラムを改造して、走るアニメーションを再生できるようにしてみましょう。「Question 1走りアニメーションを再生してみよう。」と記述されている箇所に次のコードを記述してみてください。

Lesson\_03/Player.cpp

|  |
| --- |
| void Player::AnimationControl()  {  //Question 1　走りアニメーションを再生してみよう。  if (Pad(0).IsPress(enButtonUp)) { //ゲームパッドの上ボタンが押されているなら。  PlayAnimation(AnimationRun); //走るアニメーションを再生する。  }else {  //立ちアニメーションを再生する。  PlayAnimation(AnimationStand);  }  //Question 2 ジャンプアニメーションを再生してみよう。  } |

実習課題 1(１０分)

　ゲームパッドの下、左、右が入力された時も走るアニメーションを再生することができるように、if～else文を活用してプログラムを改造しなさい。

実習課題 2(１０分)

　実習課題 1をif～else文を使用せずに、||演算子を使用することで同様の結果を得られるようにプログラムを改造しなさい。

**Lesson\_03\_02 ジャンプアニメーションを再生してみよう。**

　今度はＡボタンが押されたら、ジャンプアニメーションを再生できるようにしてみましょう。「Question 2ジャンプアニメーションを再生してみよう。」と記述されている箇所に次のコードを記述してみてください。

Lesson\_03/Player.cpp

|  |
| --- |
| //Question 2 ジャンプアニメーションを再生してみよう。  if (Pad(0).IsTrigger(enButtonA)) {  PlayAnimation(AnimationJump);  } |

どうでしょうか？正しくジャンプできましたか？キャラクターが一瞬ピクッと動きはしますが、ジャンプアニメーションは流れていないと思います。では、なぜこのようなことが起きてしまっているのか見ていきましょう。

|  |
| --- |
| void Player::AnimationControl()  {  //Question 1　走りアニメーションを再生してみよう。  if (Pad(0).IsPress(enButtonUp)) { //ゲームパッドの上ボタンが押されているなら。  PlayAnimation(AnimationRun); //走るアニメーションを再生する。  }  **11フレーム目にこのプログラムが実行されるので立ちアニメーションが再生されてしまう！！！**  else {  //立ちアニメーションを再生する。  PlayAnimation(AnimationStand);  }  //Question 2 ジャンプアニメーションを再生してみよう。  if (Pad(0).IsTrigger(enButtonA)) {  10フレーム目にＡボタンが押されてジャンプアニメーションが再生される。  PlayAnimation(AnimationJump);  }  } |

ゲームというのはパラパラアニメのようなものであるため、このプログラムはゲームが実行されている間、定期的に実行されます。そのため、10フレーム目にAボタンが押されてジャンプアニメーションを流したとしても、次のフレームではPlayAnimation(AnimationStand)が実行されてしまい、立ちアニメーションに戻ってしまいます。この問題を解決するためにはジャンプ中は立アニメーションを再生できないようにしてやる必要があります。ではプログラムを次のように改造してみてください。

|  |
| --- |
| void Player::AnimationControl()  {  //Question 1　走りアニメーションを再生してみよう。  **if (!characterController.IsJump()) { //ジャンプ中でなければ**  if (Pad(0).IsPress(enButtonUp)) { //ゲームパッドの上ボタンが押されているなら。  PlayAnimation(AnimationRun); //走るアニメーションを再生する。  }  else {  //何も押されていなければ立ちアニメーションを再生する。  PlayAnimation(AnimationStand);  }  }  //Question 2 ジャンプアニメーションを再生してみよう。  if (Pad(0).IsTrigger(enButtonA)) {  PlayAnimation(AnimationJump);  }  } |

characterController.IsJump()はジャンプ中かどうかを調べるためのものです。否定演算子！を使用していることに注意してください。否定演算子は意味を反転させるため、!characterController.IsJump()はジャンプ中じゃなければという意味になります。

**Lesson\_04\_01 キャラクターに攻撃させてみよう**

　Lesson\_04はLesson\_03の復習になります。コントローラーのボタンの入力で攻撃アニメーションを再生させてみましょう。

「Question 1 Xボタンが押されたら小攻撃アニメーションを再生する。」と記述されている箇所に次のコードを記述してみてください。

|  |
| --- |
| //Question 1 Xボタンが押されたら小攻撃アニメーションを再生する。  if (Pad(0).IsPress(enButtonX)) {  //攻撃中じゃないときにXボタンが押されたら小攻撃・  //攻撃アニメーションを再生。  PlayAnimation(AnimationAttack\_00);  //攻撃中のフラグを立てる。  attackFlag = 1;  } |

全てここまでの授業の内容の復習となります。attackFlagという変数に1を代入して攻撃中のフラグを立てているのに注意してください。この処理がないと、キャラクターは正しく攻撃アニメーションを再生することはできません。

実習課題 1(１０分)

　ゲームパッドのYボタンが押されたら中攻撃(AnimationAttack\_01)、Bボタンが押されたら大攻撃(AnimationAttack\_02)を再生できるようにしなさい。

**Lesson\_04\_02 正しく攻撃を終了させよう**

　Lesson\_04\_01の実装でキャラクターが攻撃を行うことができるようになりました。しかし、攻撃アニメーションを最後まで再生するとキャラクターが固まってしまって動作しなくなってしまいました。これは攻撃中のフラグが立っているときは、キャラクターに移動やジャンプなどが行えないように制限をかけているせいです。そのため、攻撃アニメーションの再生が終了すると、攻撃中のフラグを下す必要があります。

「Question 4 攻撃アニメーションの再生が終わったら攻撃フラグを下す。」と記述されている箇所に次のコードを記述してみてください。

|  |
| --- |
| if (!animation.IsPlay()) {  //アニメーションの再生が終わったので攻撃フラグを下す。  attackFlag = 0;  } |

これで、キャラクターは攻撃が終わると待機アニメーションに遷移して、移動やジャンプが行えるようになったと思います。