Breakthrough to Excellence!

2017-08-30
16:30–17:30

# 障害物を乗り越えるアニメーションの制御手法とその応用


代表講演者
川地克明　(株式会社スクウェア・エニックス　テクノロジー推進部)
共同開発者
青野孝悠　(株式会社スクウェア・エニックス　第2ビジネス・ディビジョン)
小倉孝典　(株式会社スクウェア・エニックス　第6ビジネス・ディビジョン)
David Goodhue　(株式会社スクウェア・エニックス　テクノロジー推進部)


CEDEC2017
Computer Entertainment Developers Conference

SQUARE ENIX®

©2017 SQUARE ENIX CO., LTD. All Rights Reserved

# 障害物を乗り越える動作

FINAL FANTASY XV, SQUARE ENIX, 2016

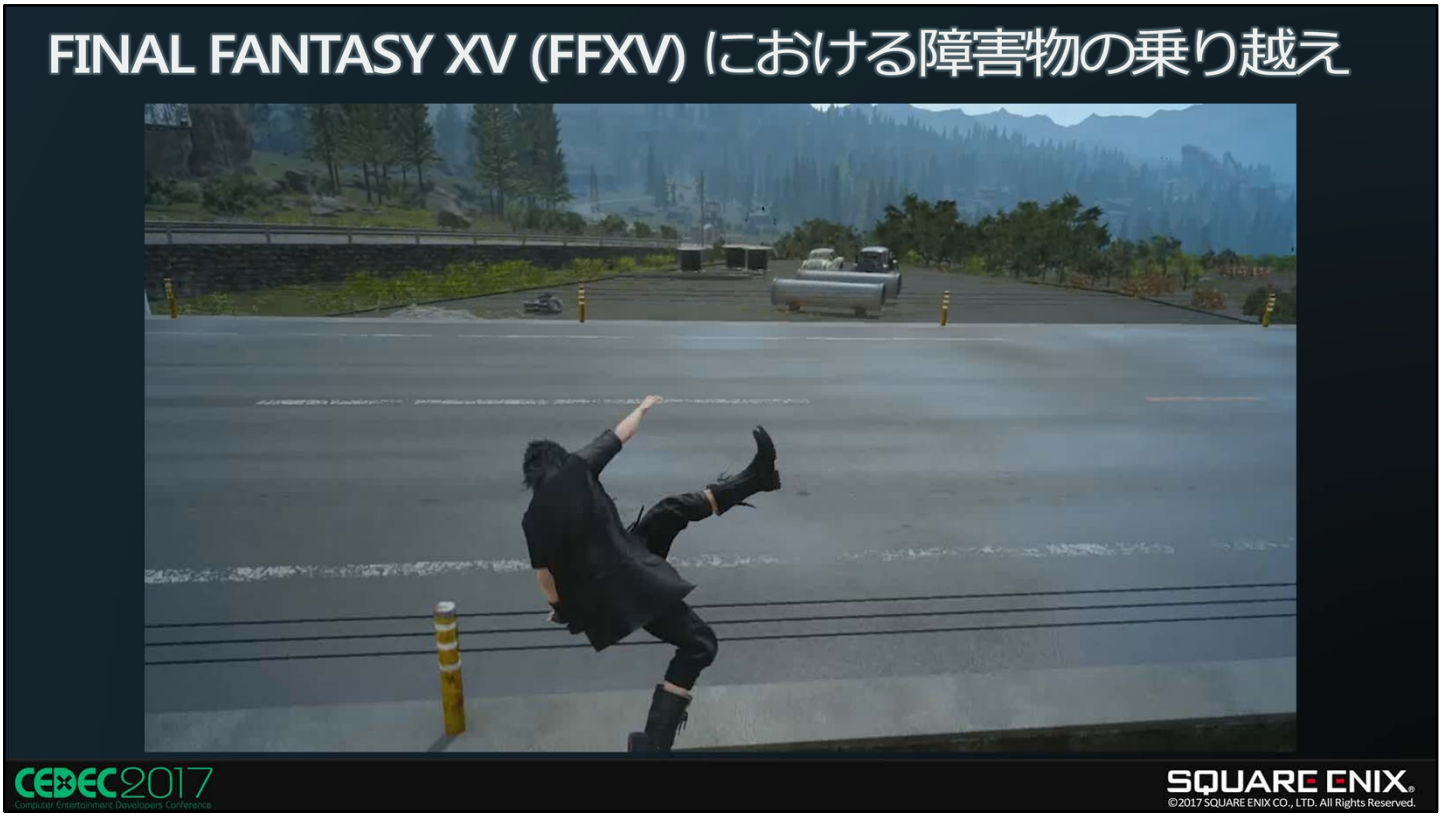
- 進路上にある障害物を自動的に乗り越え、移動状態を継続させる
  - 近年のアクションゲームではすでに一般的なフィーチャ
  - 機能の呼称としては “vaulting”, “parkour” など


SQUARE ENIX
©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

この発表では、アニメーションシステムで障害物を乗り越える動作の実装について説明します。

この機能は、キャラクタの進路上にある障害物を検知して、自動的に乗り越える動作に切り替えることによって、キャラクタが移動している状態を継続させるものです。

近年のアクションゲームではごく一般的な機能となっています。




本発表の中で紹介する乗り越え動作に関する実装は、昨年 (2016年11月) 発売した Final Fantasy XV のアニメーションシステムの一部として開発したものです。

FF15 では、プレイヤキャラクターを柵などの障害物にむかって移動させると、自動的に柵を乗り越える機能が実装されています。

このとき、柵を乗り越える動作は柵に近づくと自動的に再生されるので、プレイヤーはジャンプの操作を行う必要はありません。

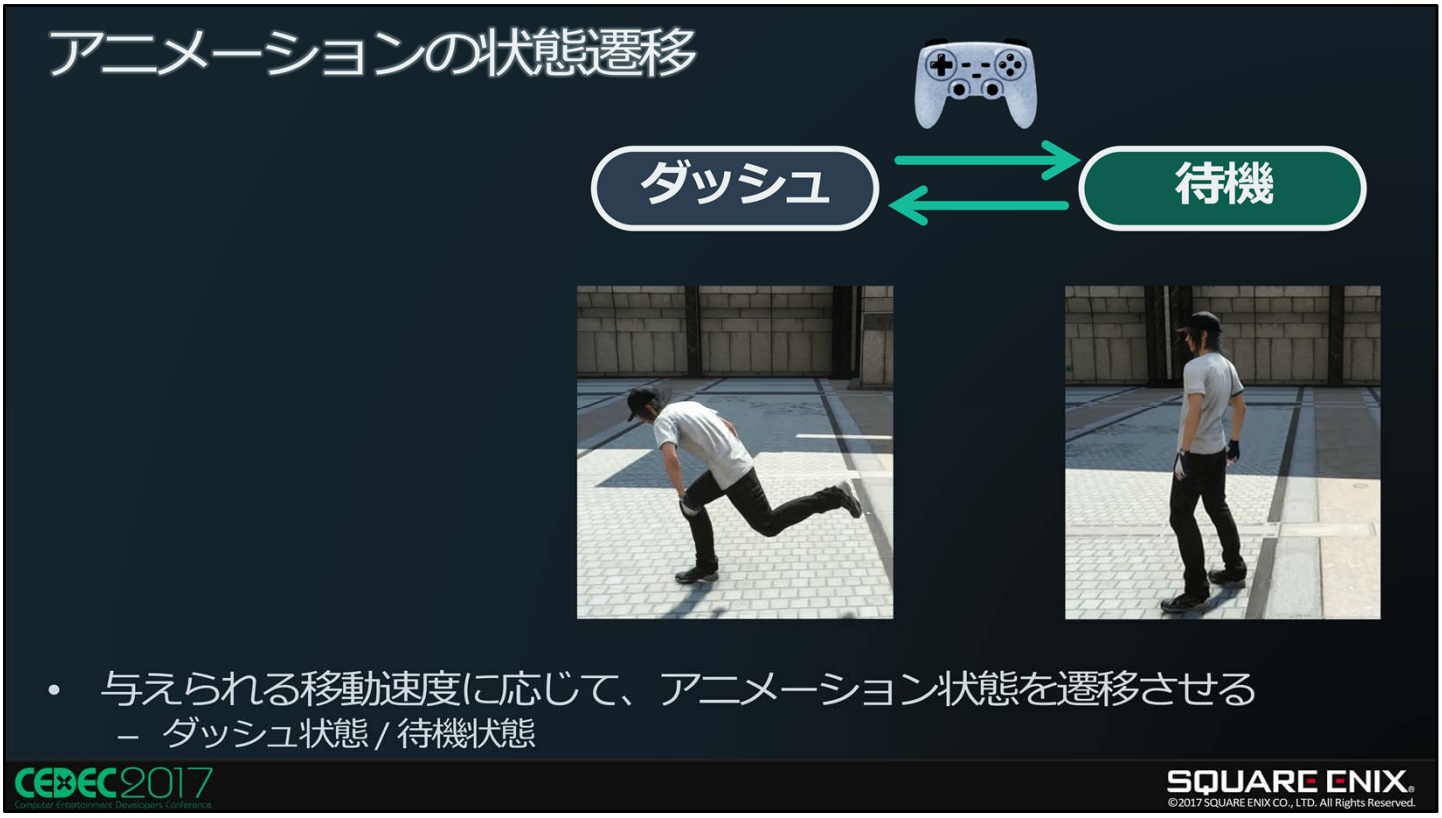
また、プレイヤーの移動するスピードや、柵の高さが異なる場合には、これを検知して何種類かの乗り越え動作が再生されるようになっています。

プレイヤーが柵を越えて移動するとき、パーティの仲間キャラクタもプレイヤーを追いかけてくるので、同じように乗り越え動作を再生することで柵を越えて移動します。

この障害物を乗り越える仕組みは、他にもいろいろなキャラクタで利用されています。

プレイヤーと仲間は乗り物となるモンスタ (チョコボ) に乗って移動することができます。このとき、チョコボも仲間のキャラクタと同じように、柵を自動的に飛び越えて移動する仕組みが実装されています。

敵のキャラクタについても、プレイヤーを追いかけるように移動するときに、間に障害物があれば、これを乗り越える動作を自動的に行います。

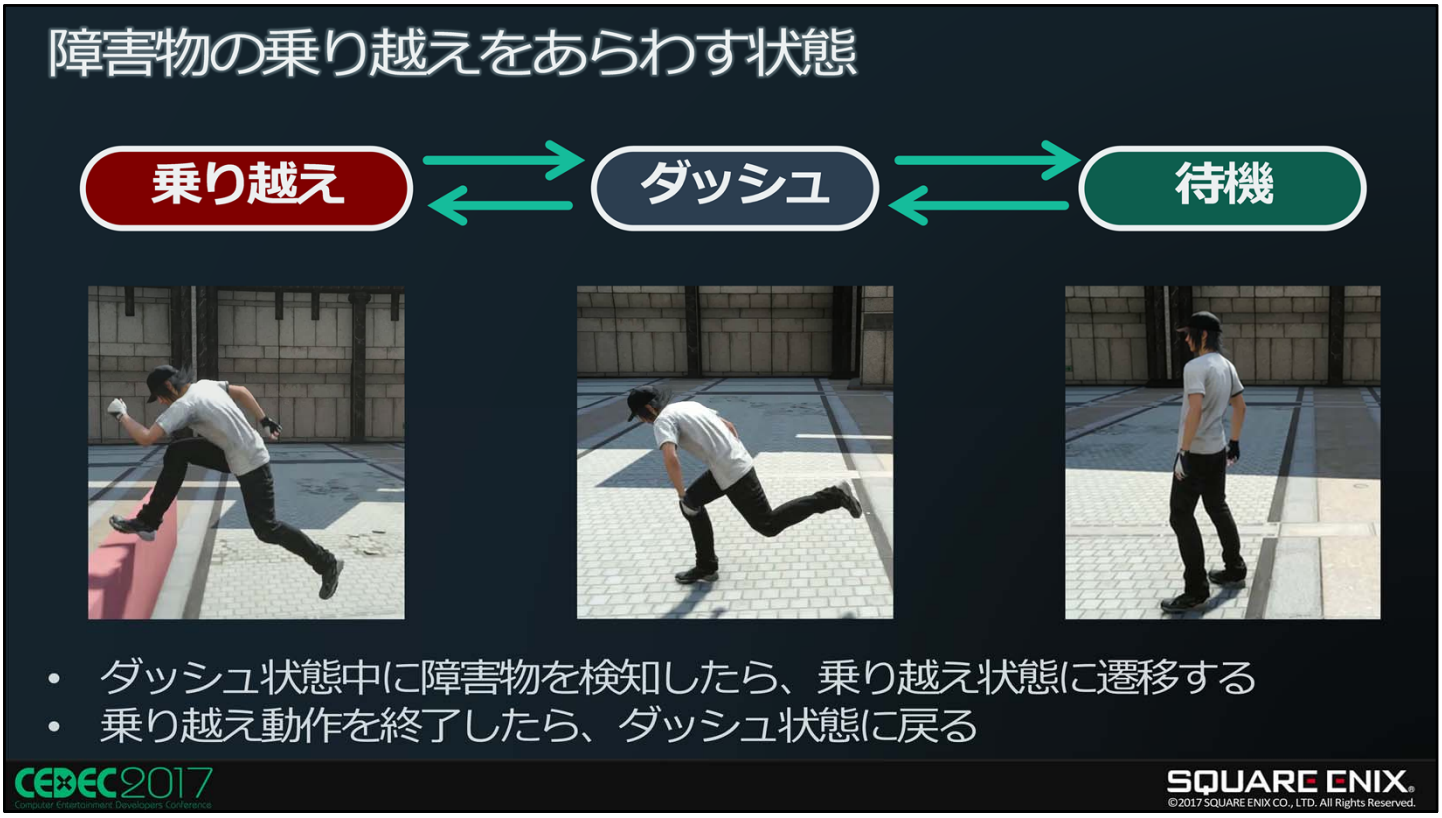



FFXV のアニメーションシステムは、状態遷移とブレンドツリーによってキャラクタの動きを定義しています。

典型的なアニメーションの状態遷移は、速度によってアニメーションの状態をスイッチさせるようなグラフとなります。

速度が0のときは待機状態、速度が0よりも大きい時はダッシュで移動する状態に対応します。

このようにして、ダッシュと乗り越えという動作の状態を含むアニメーションを定義しています。



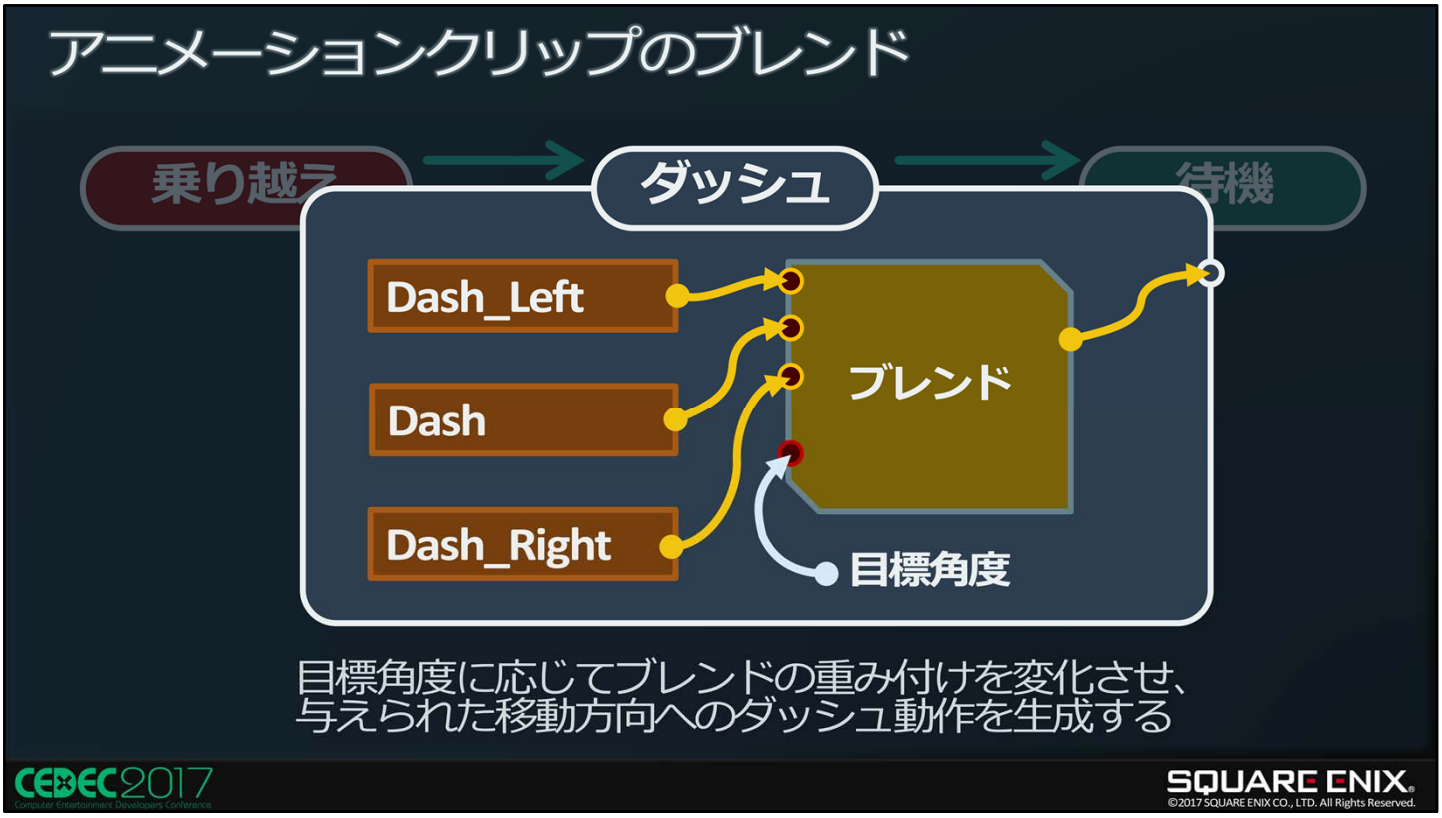

アニメーション状態遷移の一部として乗り越え動作を定義すると、ダッシュ状態にあるときに障害物を検知したら、乗り越え状態に遷移するような状態遷移図となります。

乗り越え状態のアニメーションを再生し終わったら、ふたたびダッシュ状態に戻ります。

次に、それぞれの状態、ダッシュや乗り越えなどの状態の内部がどのように実装されるかについて見ていきます。

アニメーション状態のいちばんシンプルな定義は、アニメーションクリップのデータを再生した動きをそのまま出力とするものです。

このとき、ダッシュのアニメーション状態にはアニメーションクリップが一つ含まれていて、ダッシュで直進するだけの状態となっています。

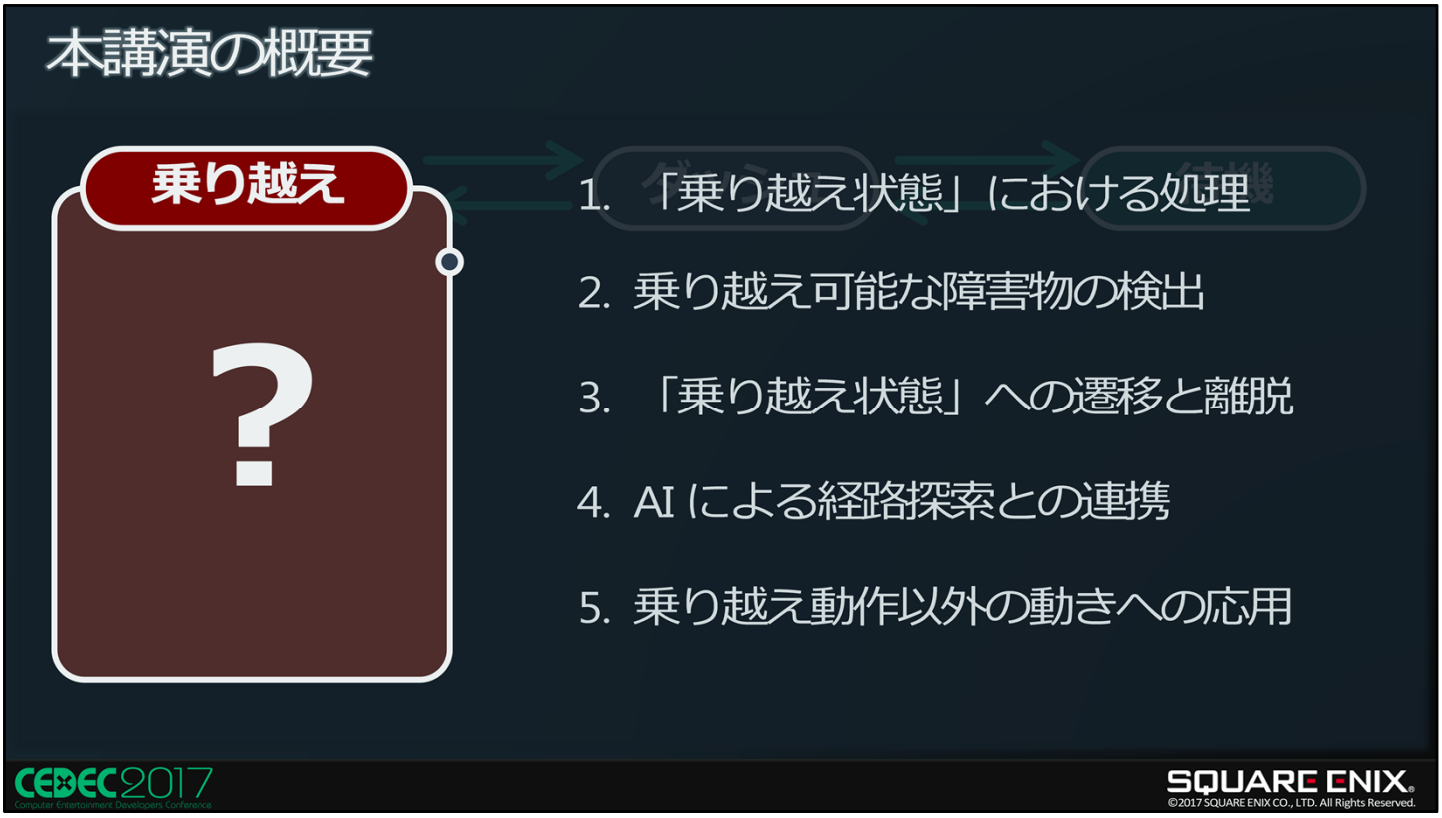



ダッシュしながら左右にカーブする状態に拡張する場合は、もう少し複雑な状態の定義が必要になります。

実装方法はいろいろありますが、このスライドの例では、まず用意するダッシュのアニメーションクリップを3種類に増やします。

3種類のデータは直進、左カーブ、右カーブに対応したデータです。

この3つのクリップを、目標とするカーブの角度に応じてブレンドすると、ダッシュしながら左右にカーブする動きを出力する状態を定義することができます。



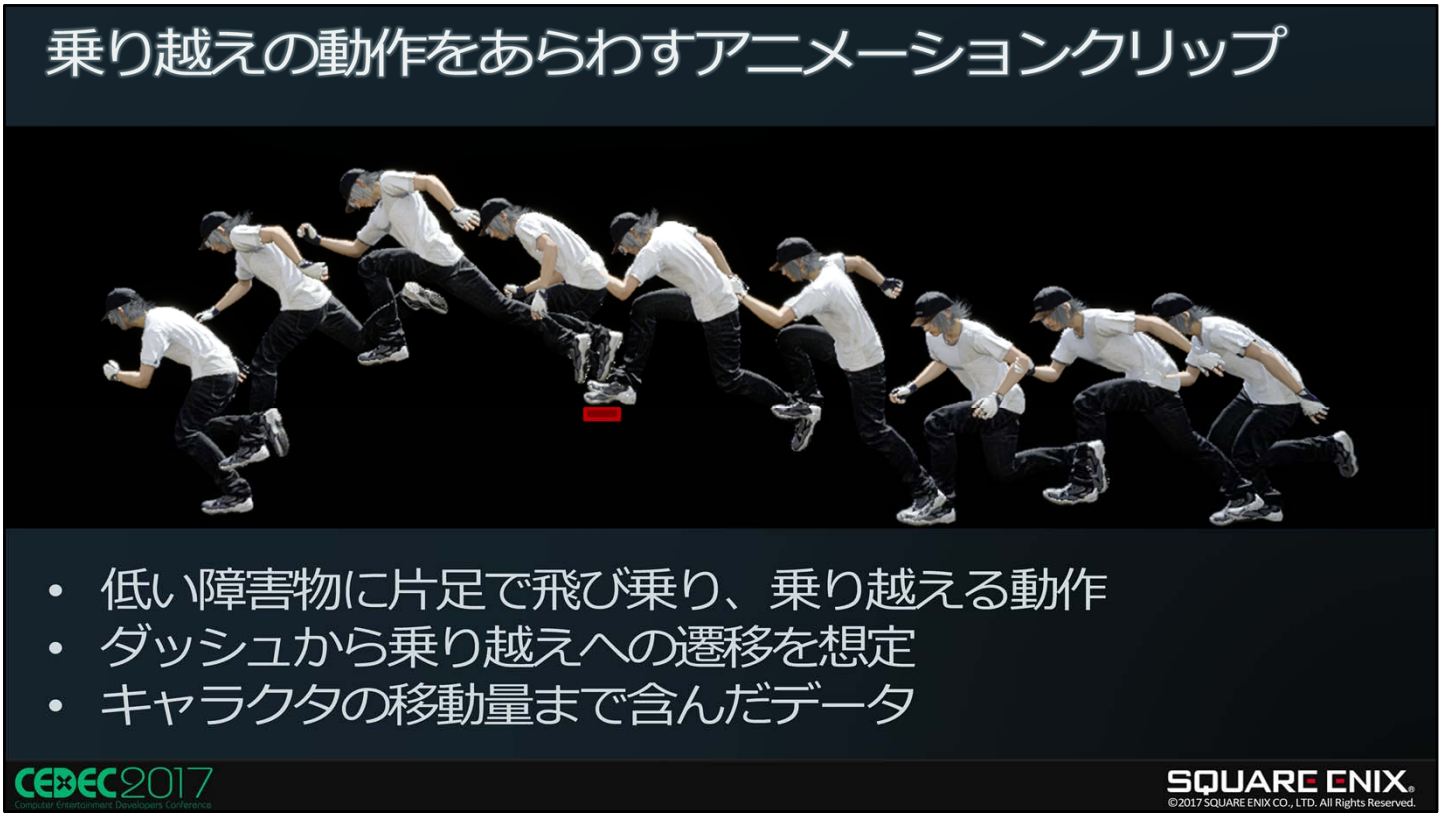

このように、アニメーションの一つ一つの状態については、アニメーションクリップを再生したり、ブレンドするような処理によって実装されていますが、本発表では、障害物を乗り越える動作の状態はどのような処理によって実装されるかということについて説明したいと思います。

基本的には、乗り越え状態での処理は、障害物に応じてアニメーションクリップを調節するような処理になりますが、本発表ではまず、このような調節処理の設計と実装について、FFXV での実例に基いて説明します。

本発表の残りの部分では、障害物を検出するために必要なデータを定義する方法や、アニメーションと連携するAIなどのシステムとの接続方法について説明します。

また最後に、このようにして実装した乗り越え動作を制御するアニメーションシステムをより一般化して、乗り越え動作以外の動きに応用した事例についても紹介します。

ではまずはじめに、アニメーションが乗り越え状態にあるときの処理について説明します。

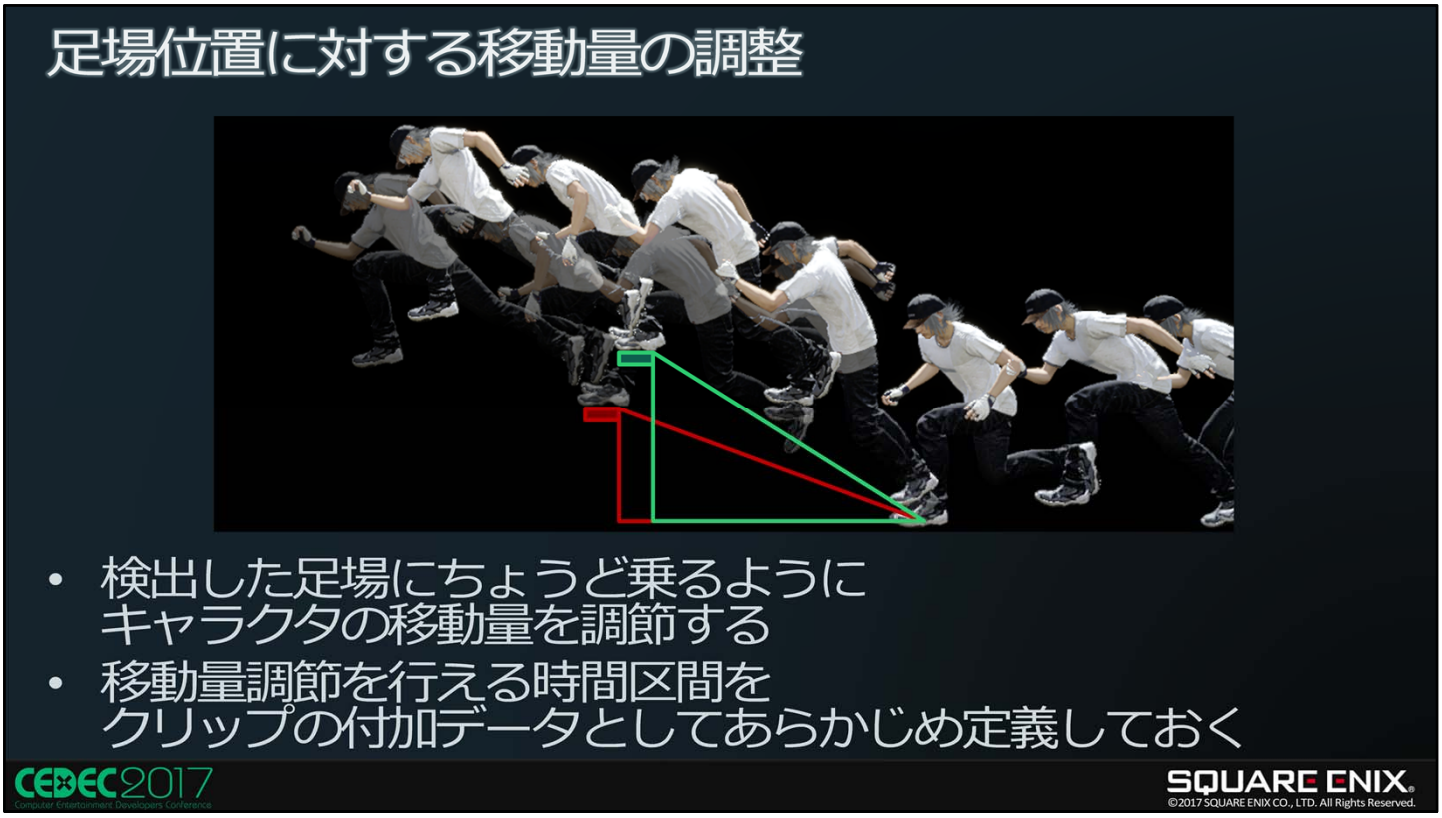



このスライドは、乗り越えの動作を表すアニメーションのデータ、アニメーションクリップを横から見た様子をあらわしています。

この乗り越え動作は、低い障害物に片足で飛び乗って乗り越える動作です。ダッシュ状態から遷移したときに、ちょうどなめらかに接続できるような想定で作成されています。

このデータを再生することで障害物を乗り越える動きを生成しますが、実際のマップにはいろいろな障害物が配置されるので、個々の障害物に対して動きを調節する必要があります。

アニメーションデータにはキャラクタの移動量が含まれているので、この移動量を調節することで障害物に対応していきます。




このスライドでは、もともとアニメーションデータとして赤い足場に乗るように作られている乗り越え動作があった場合に、障害物の高さがもっと高い位置にあることがわかったので、その緑色の足場位置に合うように調節している様子を表しています。

アニメーションの状態としての乗り越え動作の状態で行う処理は、基本的にはこのようなシンプルな調節の処理となっています。

乗り越え動作のアニメーションクリップに対しては、クリップのどの部分で調節処理を行えばよいか、ということをあらかじめ定義しておきます。

# 移動方向の調整

- 障害物に正対するように移動方向を補正する
  - 方向補正を行う時間区間を、クリップに対する付加データとして定義しておく

CEDEC2017
Computer Entertainment Developers Conference
SQUARE ENIX
©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

また、もし必要がある場合には、キャラクタの移動方向を補正します。

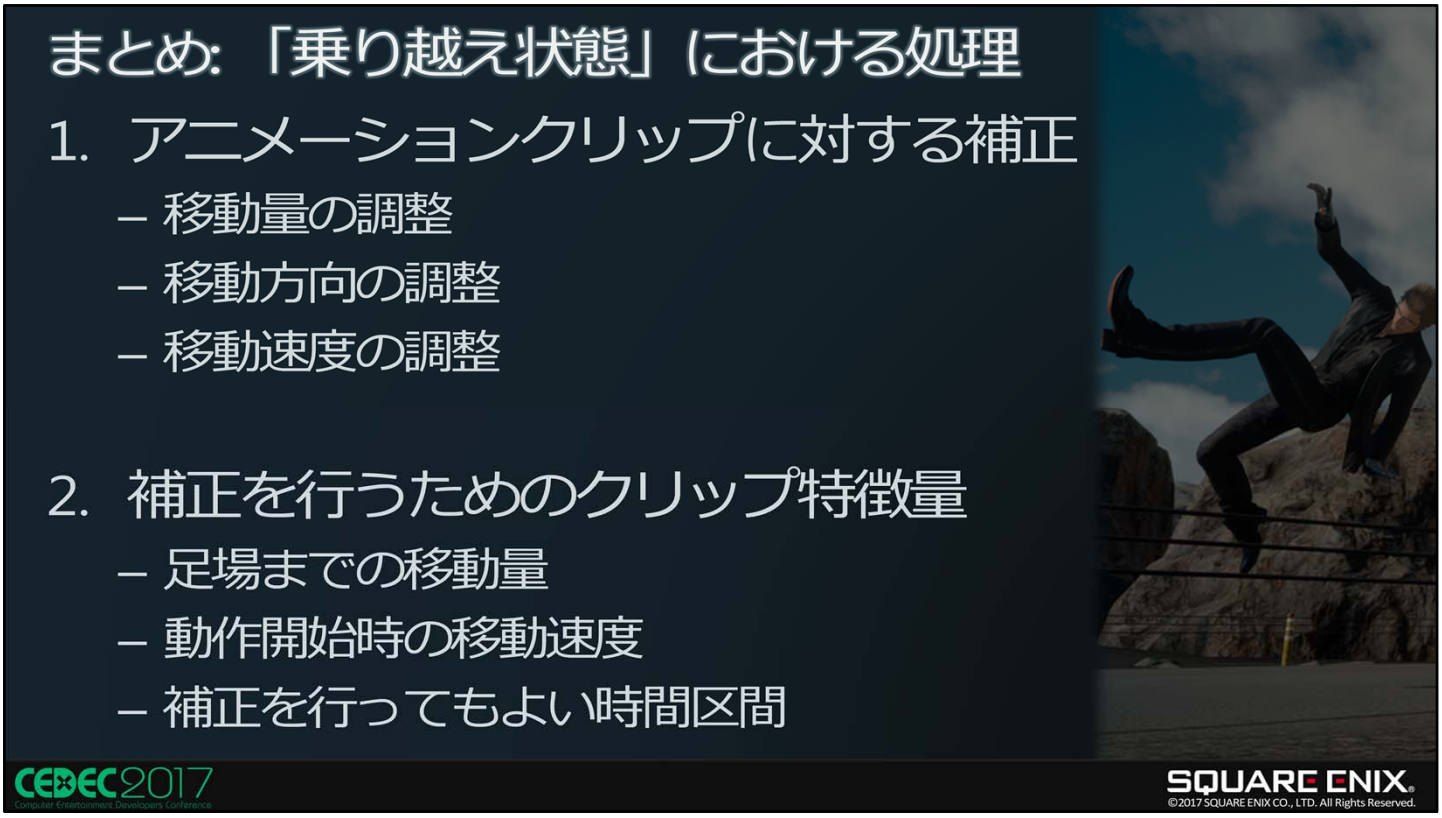
これは、乗り越え動作を行っている時に、キャラクタが障害物と正面から向き合うような方向に補正を行う処理です。

この処理についても、補正を行う時間区間をアニメーションクリップに対して定義しておきます。

乗り越えに遷移する直前としてはダッシュ状態を想定していますが、実際のダッシュ状態は一定の速度でダッシュしているわけではなく、状況によって速度のばらつきがあります。

このようなさまざまなダッシュ速度から、乗り越え動作に遷移したときに速度が不連続にならないように、乗り越え動作の移動速度に適切な係数を掛けあわせる操作を行います。

このような補正により、アニメーションの状態遷移の前後で不連続に速度が変化するのを防ぎます。




以上が、乗り越えの状態でアニメーションクリップに対して行う処理になります。

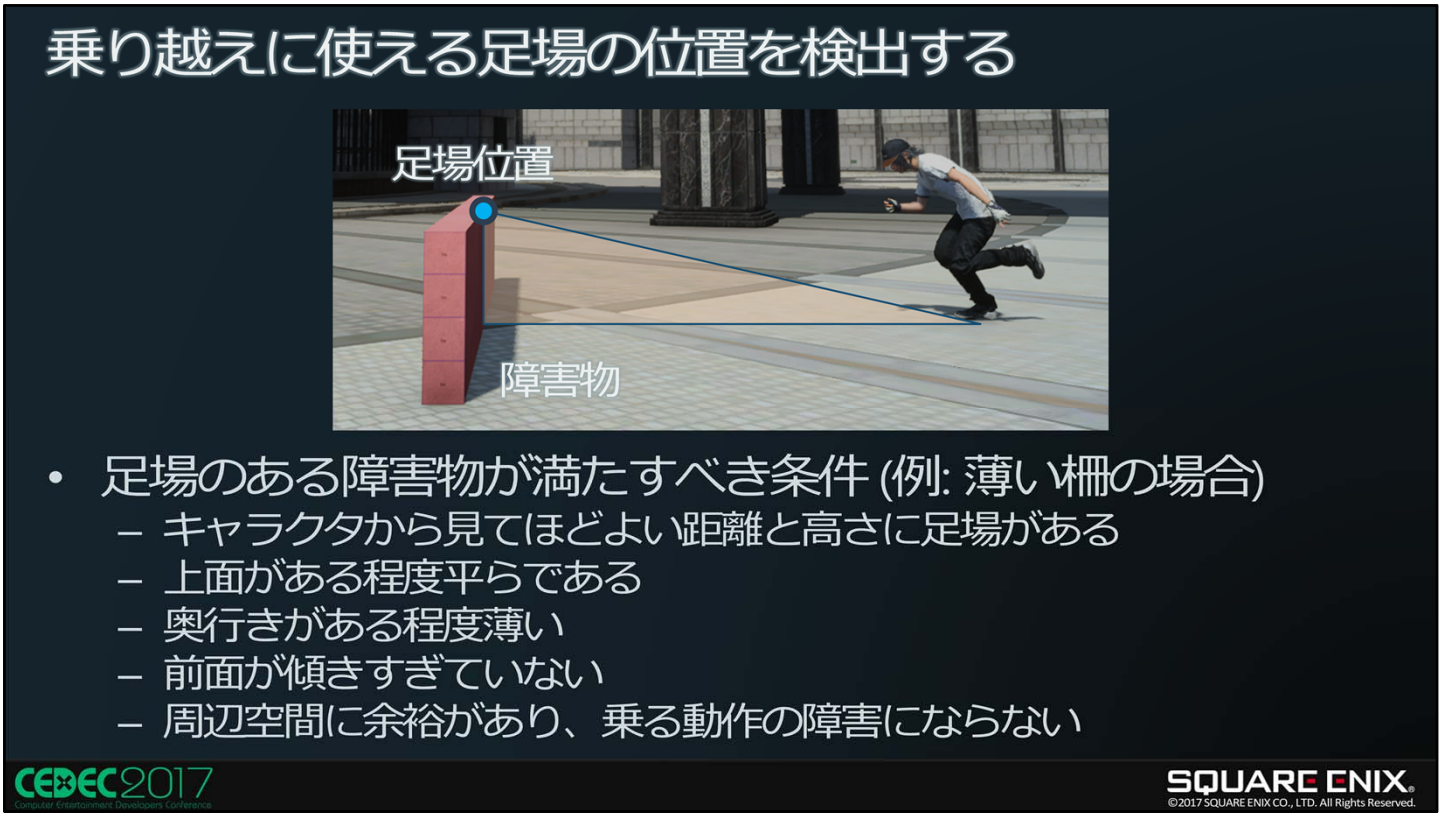
処理の内容としては、障害物に対する位置関係の補正や、移動速度の連続性に関する処理を行います。

またそのために、アニメーションクリップに対する特徴量データとして、補正を行ってもよい時間の区間や、足場との位置関係、移動速度といったデータをあらかじめ与えておきます。

これらの処理を実装することで、キャラクタが障害物を乗り越える基本的な処理は完成です。

発表の残りのスライドでは、こういった補正処理をシステムにどのように組み込むかということについて説明します。

まず、補正を行うために、マップ上に配置された障害物を検出する手続きについて説明します。




乗り越え動作を生成するためには、障害物の上にキャラクタが着地する足場の正確な位置を決める必要があります。

このような足場は、配置や形状について幾つかの条件を満足している障害物上の点となります。スライドに図示している薄い壁のような障害物の場合には、

・キャラクタからの距離がほどよく
・壁の上の面がある程度平らで、
・壁の進行方向の奥行きがある程度薄く、
・壁の前の面が傾いておらず、
・足場の周辺の空間に余裕があって、乗り越え動作の障害にならない

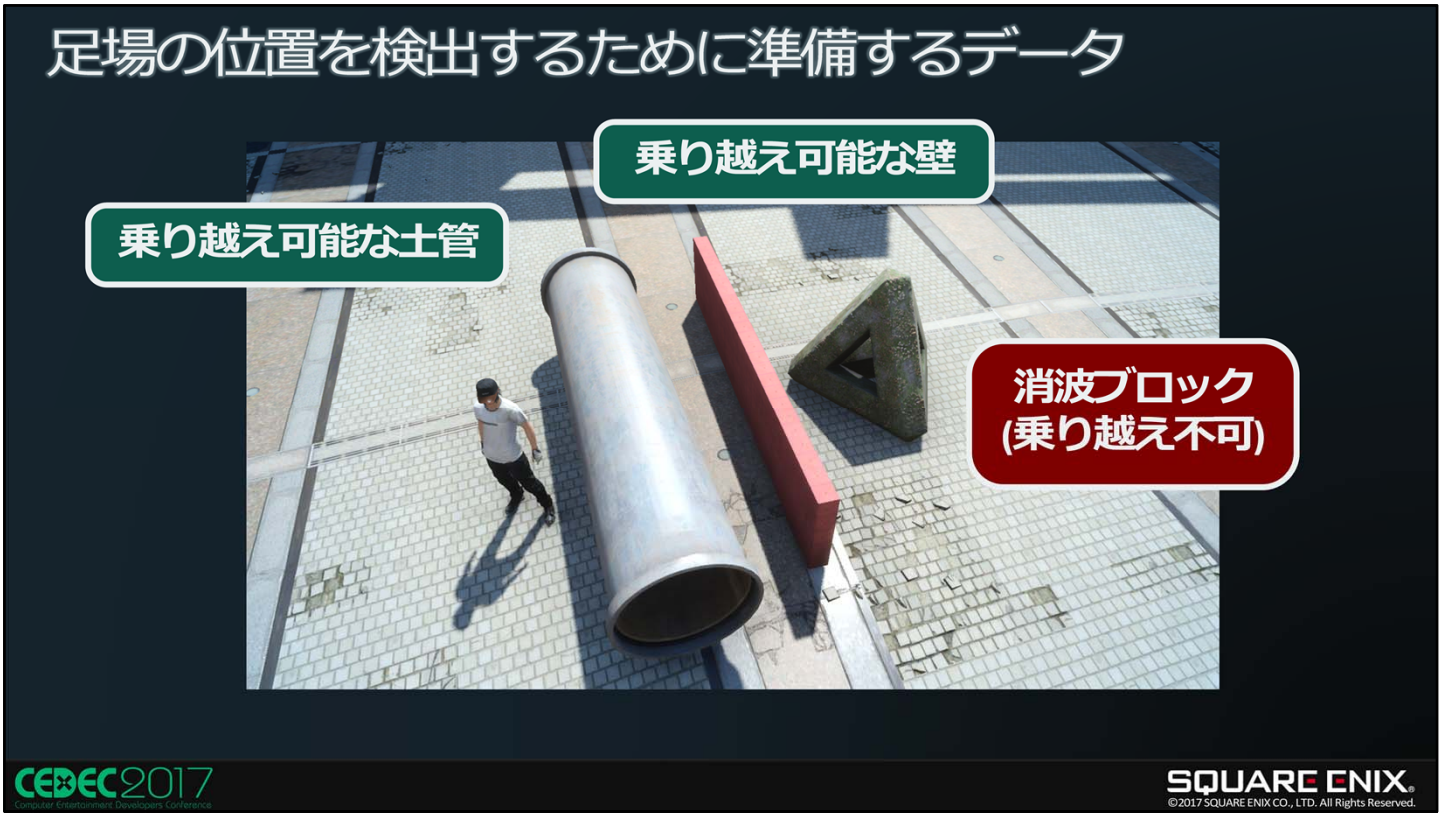
といった条件が必要です。

反対に、乗り越え動作の実行に適当ではない形状は、スライドのように

上面が傾きすぎていたり、
そもそもの高さが低すぎたり、
壁の上に障害物があったり、
壁の前の面が傾きすぎていたり

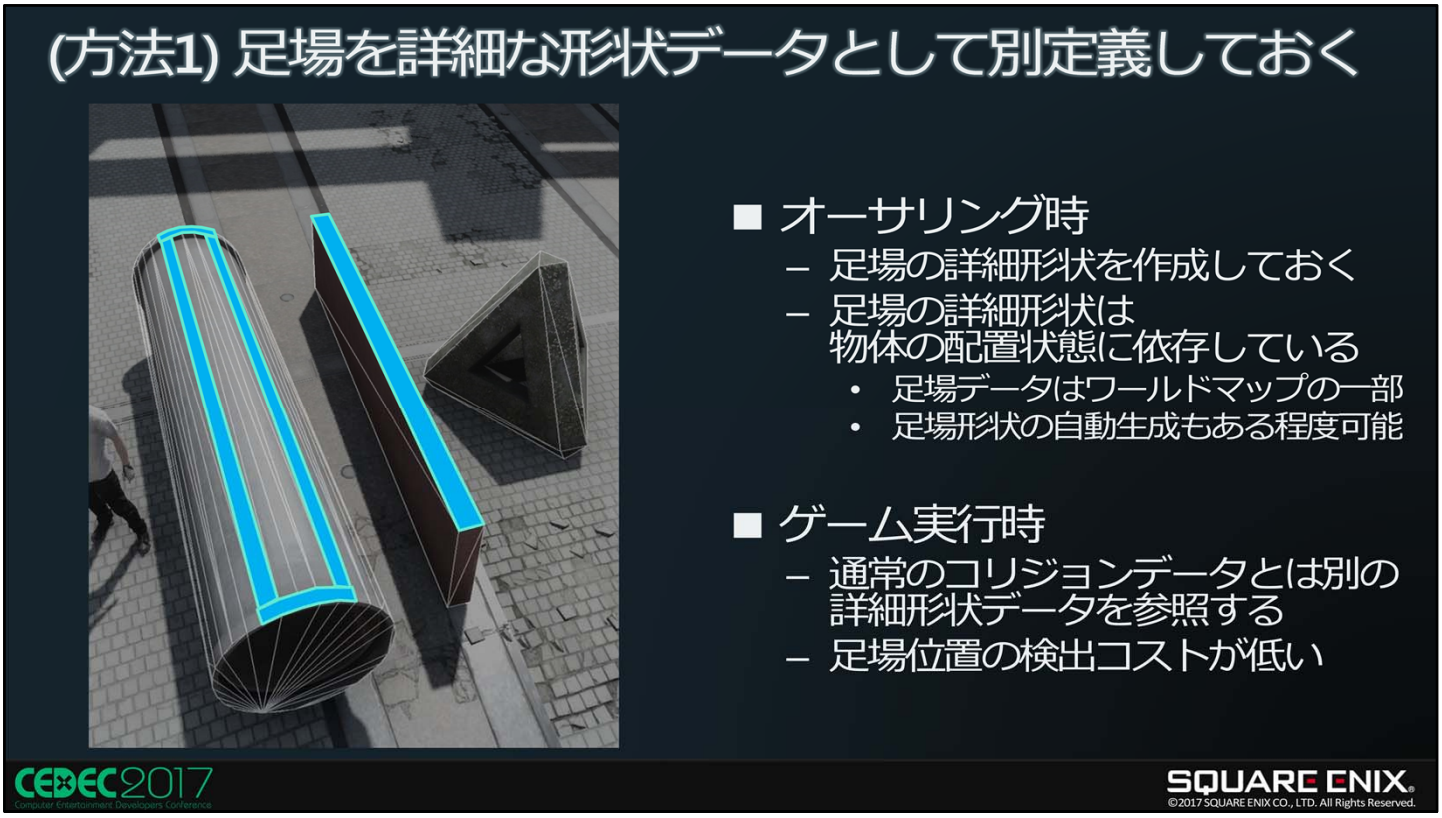
といったものとなります。

足場の位置を決める時は、このような不適当な形状をはじいていく必要もあります。




障害物の上で足場の位置を決めるためには、そのためのヒントとなるデータを用意しておきます。

ここでは例として、乗り越られる壁、乗り越えられる土管、乗り越えられない三角形の消波ブロックが配置してあるシーンを考えます。



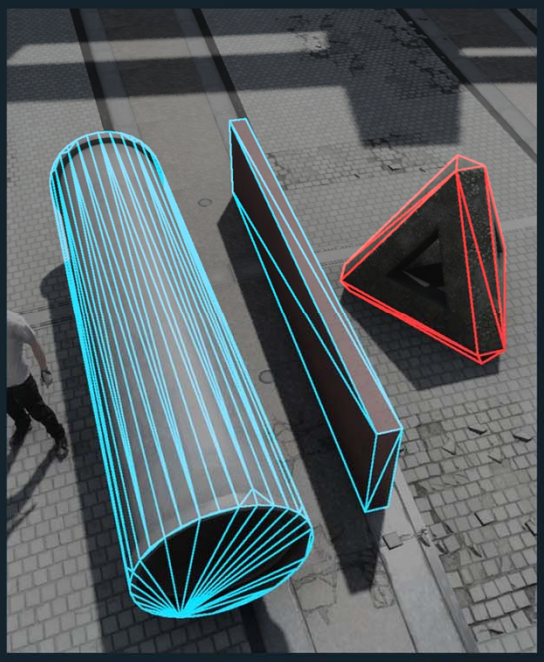

足場の位置を決めるためのデータの持ち方は大きく分けて2つの方法があるといえます。

一つ目の方法は、足場として乗れる場所を別のデータとして作成しておく方法です。スライドでは壁の上や土管の上に水色の領域がありますが、この水色の部分が乗り越え動作の足場となる部分を示しています。

ゲームを実行するときは、この足場データを参照することで、キャラクタが使える足場を効率よく検出することができます。

一方で、このような足場のデータは障害物の配置に依存しているので、ワールドマップの一部となります。つまり、障害物をワールドマップに配置して、その後で何らかの方法で足場データを作成していくことになります。

## (方法2) 物体単位で乗り越えの可否を定義しておく



- ■ オーサリング時
  - 乗り越えできるかどうかを物体単位で決めておく
  - 物体の配置を気軽に変更できる
- ■ ゲーム実行時
  - 物体上の足場位置を動的に決定
  - 通常のコリジョンデータを利用

→ FFXV ではこちらの方法2を使用

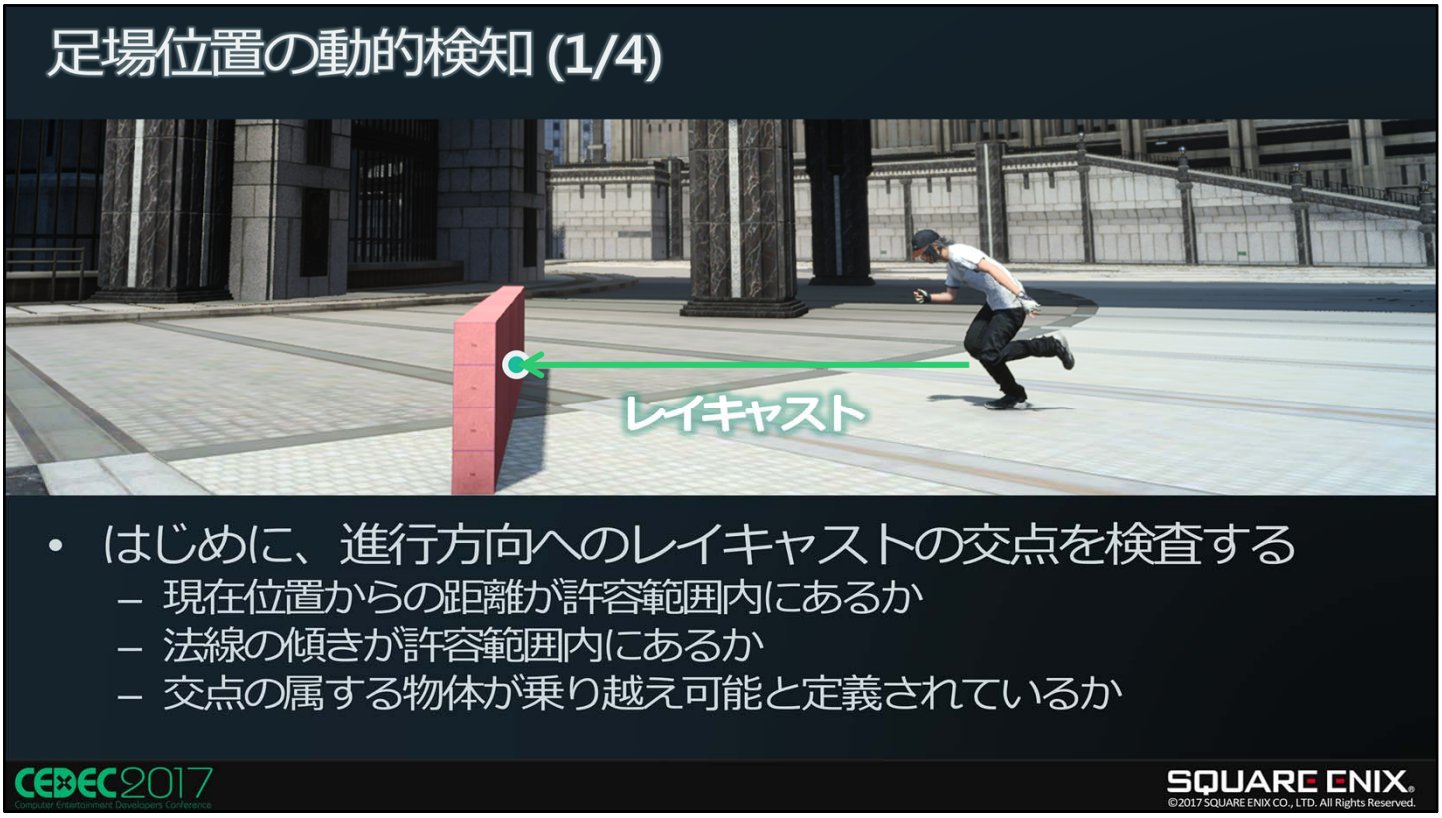
CEDEC2017 Computer Entertainment Developers Conference
SQUARE ENIX ©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

もう一つの方法は、乗り越えられるかどうかを障害物ごとに属性として決めておく方法です。

詳細な足場データを別のデータとして持たないので、データ作成のコストはかかりません。

その代わり、実際に障害物の上に足場として使える場所があるかどうかを、ゲームの実行時に動的に検査していくことになります。したがって、足場を検出する計算コストが発生します。

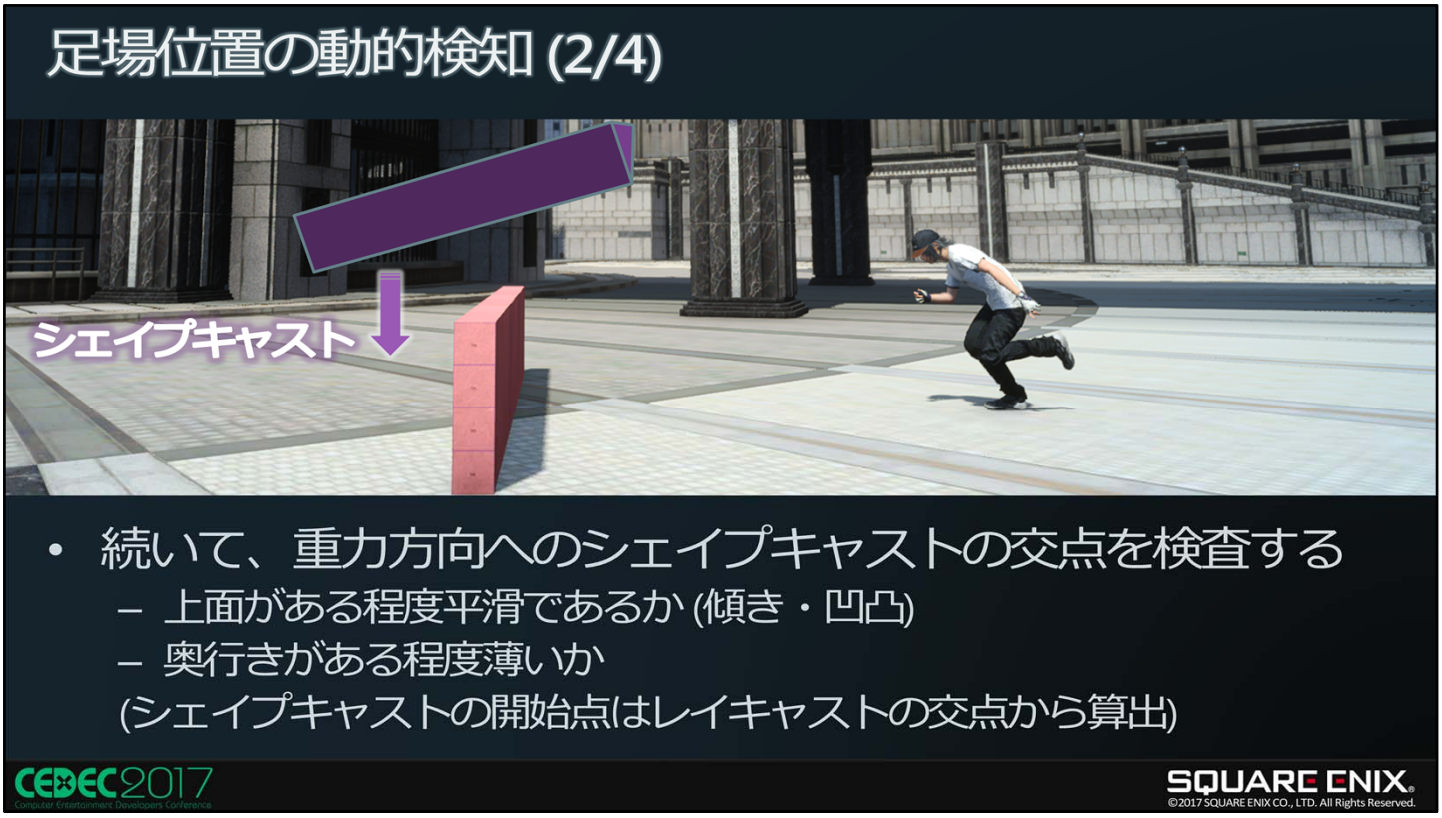
FFXV における乗り越え動作の制御では、こちらの方法を採用しています。次のスライドからは、ゲームの実行時に詳細な足場の位置を検出する方法について説明します。




足場の位置を検知するために、まずはじめに、キャラクタの進行方向にレイキャストを行います。

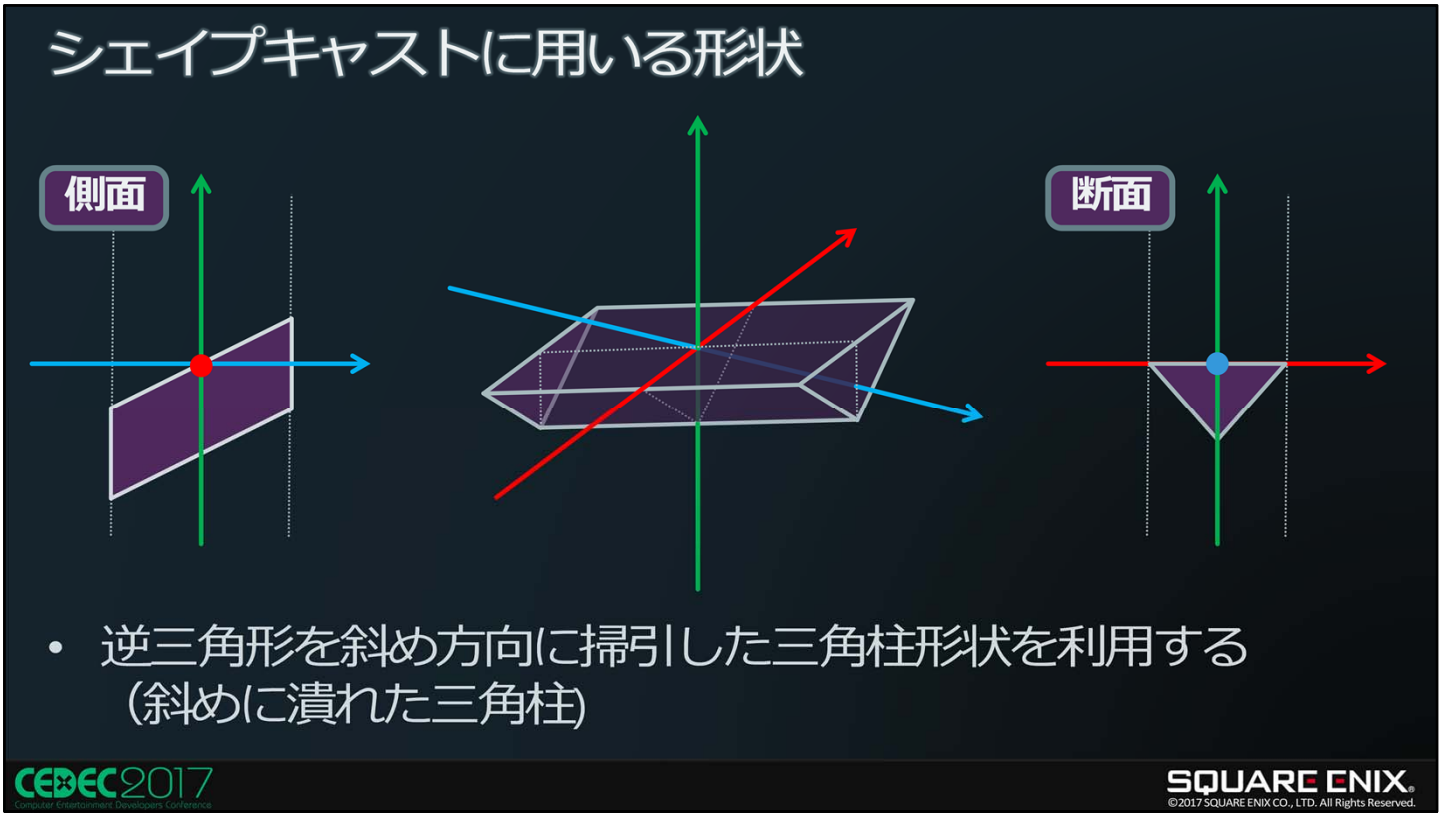
その結果、乗り越え可能の属性を持つ障害物と交点があれば、次のステップにすすみます。

このとき、交点の位置がキャラクタの現在位置からみて適当な距離の範囲にあるか、また、交点の法線の傾きが許容される範囲にあるかといった条件を検査します。



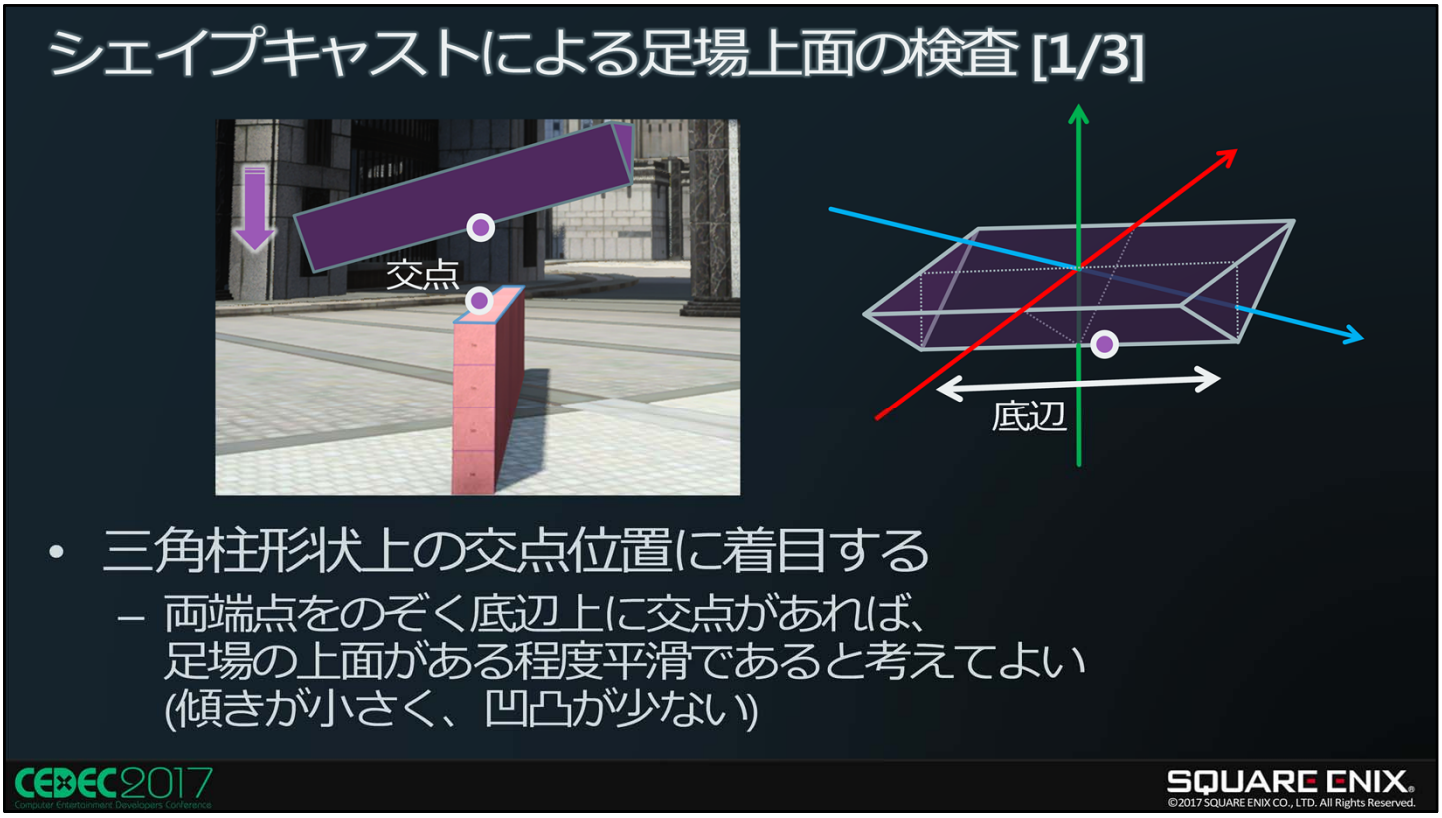

次に、レイキャストで得た交点の位置にもとづいて、下方向にシェイプキャストを行います。

このシェイプキャストによって、足場の上の面がある程度平らになっているかどうか、また、進行方向への足場の奥行きがある程度薄いものかどうかという条件を検査することができます。



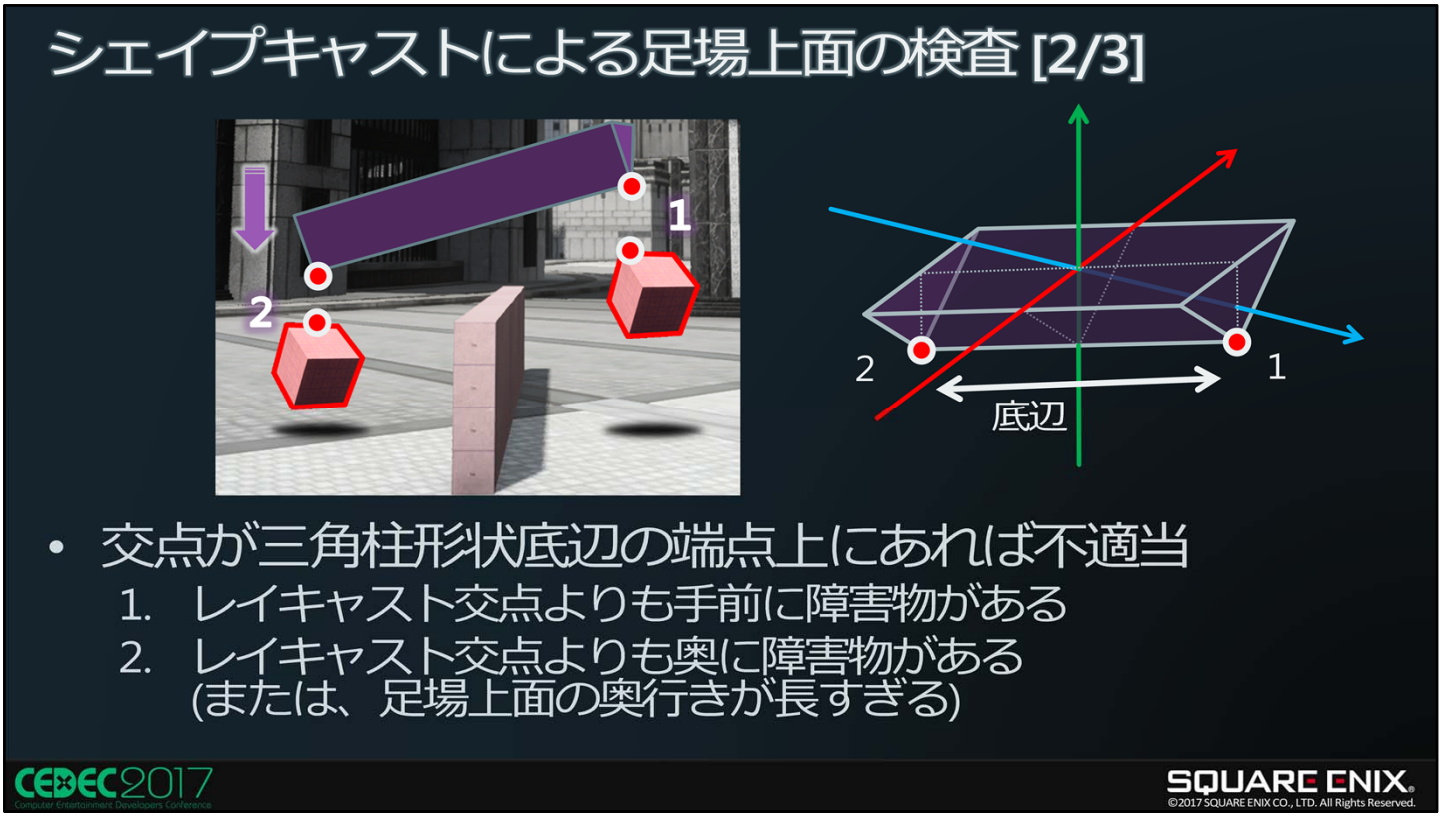

このような検査に用いるシェイプキャストの形状は、三角柱を斜めに潰したような形状になっています。

三角柱の尖っている部分が下になっており、この尖った部分を障害物にぶつけるように下方向にシェイプキャストすることで、足場に関する条件が満たされているかどうかを検査します。




障害物の上面がある程度平らであるかどうかは、三角柱によるシェイプキャストの交点位置に着目することによって検査することができます。

もし交点が三角柱の底辺の上にあれば、障害物の上の面はある程度平らであると考えてよいことになります。



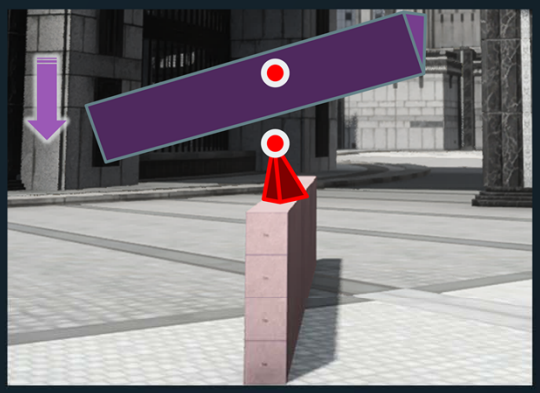

一方で、交点の位置がそれ以外の場所にある場合、このスライドで示すように、底辺の内部ではなく、底辺の端点が交点になっている場合には、レイキャストで検出した交点以外に、乗り越え動作の障害となる物体があると考えられます。

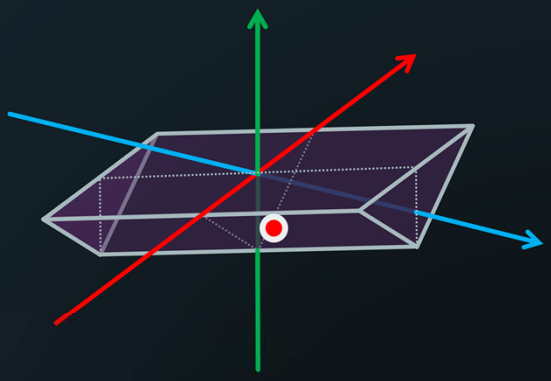
もし三角柱の手前側 1 の場所に交点があれば、レイキャストの交点よりも手前に何らかの障害物があると考えられます。

また、三角柱の奥側 2 の場所に交点があれば、レイキャストの交点よりも奥に何らかの障害物があると考えられます。

これらの場合は、レイキャストで検出したものとは別の障害物によって乗り越え動作を実行することができないと判定できます。

## シェイプキャストによる足場上面の検査 [3/3]



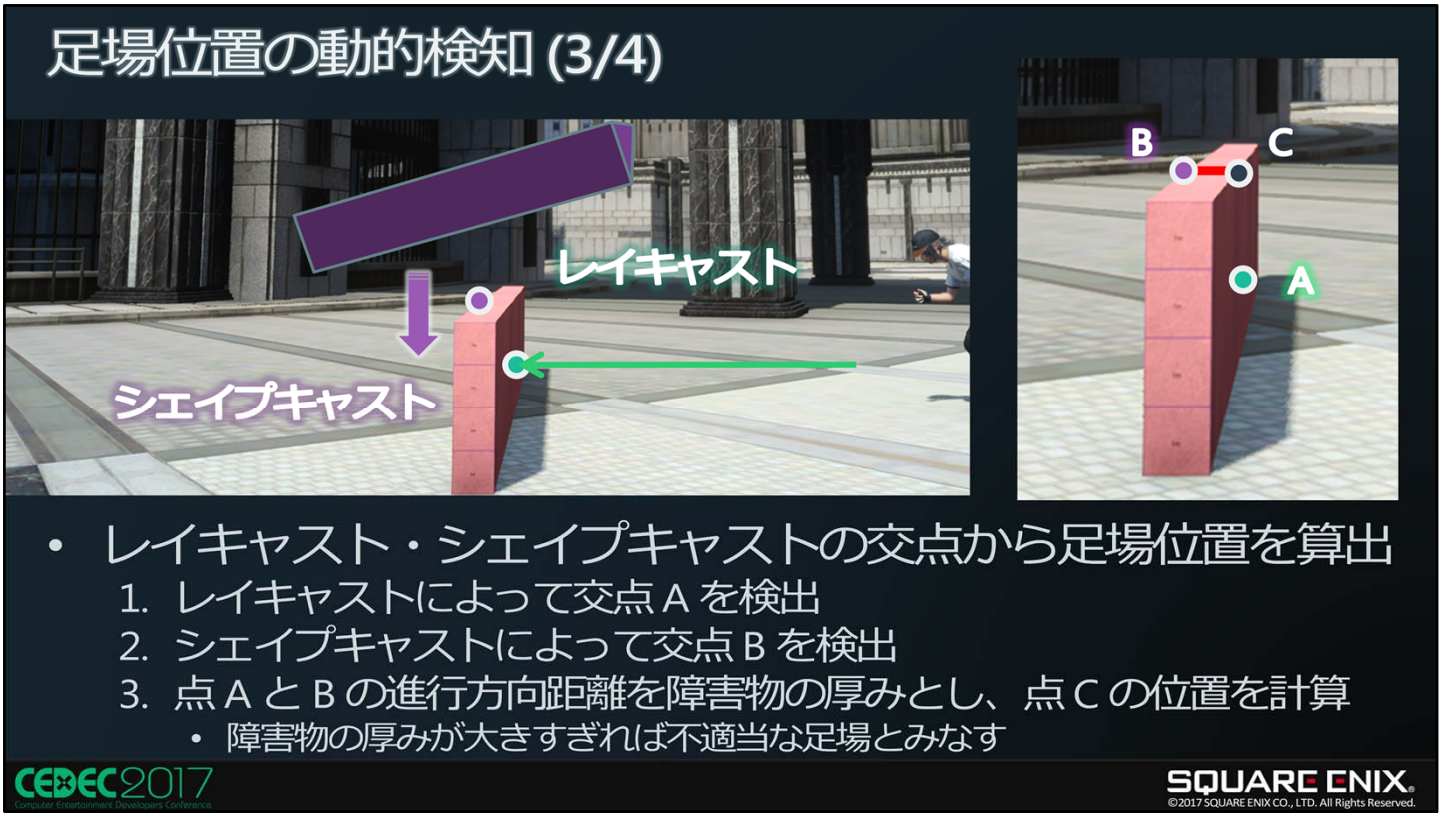


- 交点が三角柱形状の底 (側面) にあれば不適当
  - 足場上面に大きすぎる凹凸がある
  - 足場上面が傾きすぎている

CEDEC2017
Computer Entertainment Developers Conference
SQUARE ENIX
©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

また、シェイプキャストの交点が、三角柱の側面にあった場合には、障害物の上の面に何らかの凸凹があると考えられます。

障害物の上の面の傾きが大きすぎるときもこのような状況になるので、これらの場合には適当な足場が存在しないと考えられます。

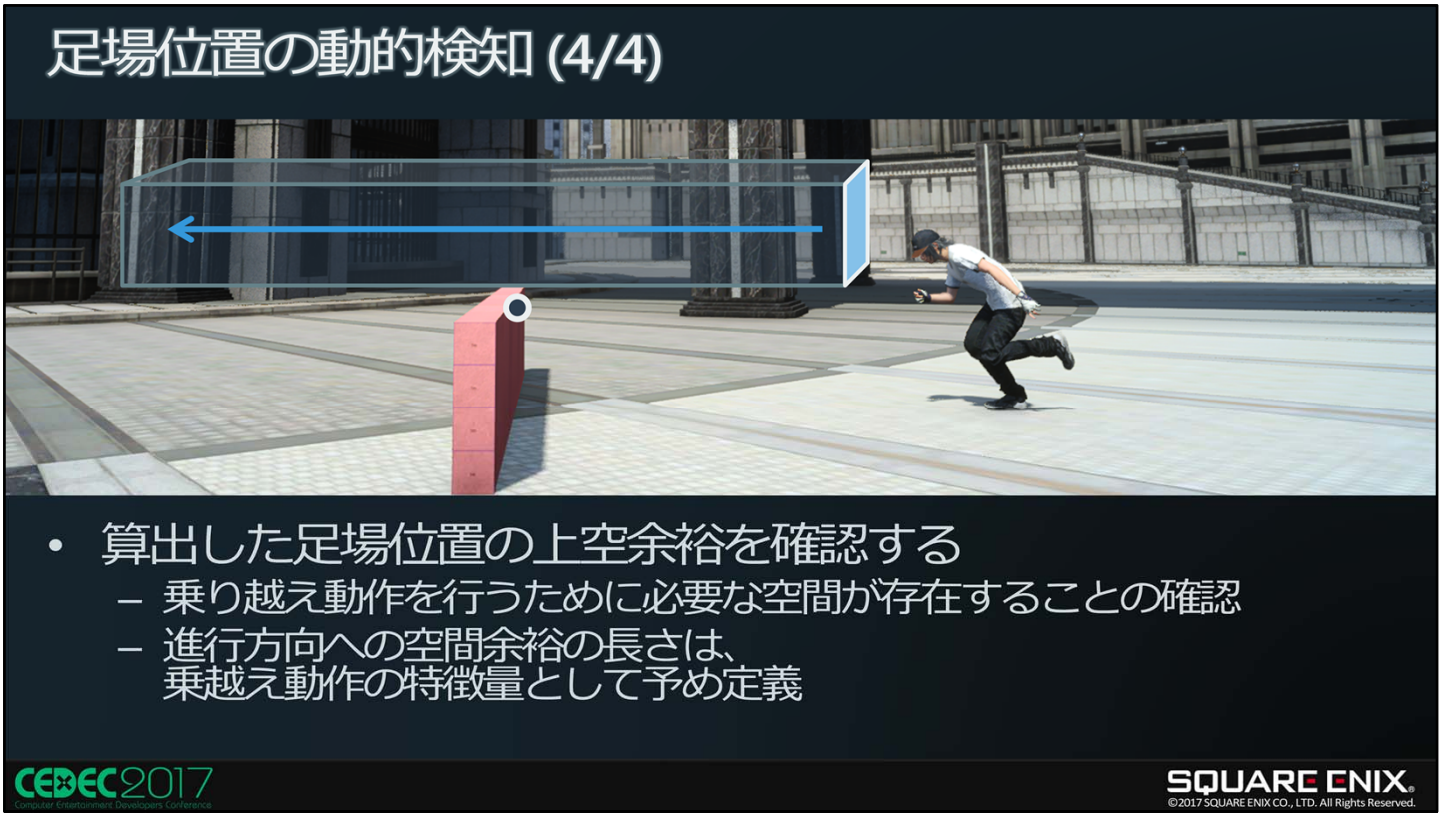



こうしてレイキャスト・シェイプキャストの2つの交点a, b を調べることで、障害物の上の面がある程度平らであることが確認できます。

この2つの交点ab は進行方向に対してズレていることになりますが、このズレが障害物の厚みを表しています。

厚みを考慮することで、スライドで点Cとして示す足場の位置を計算することができます。

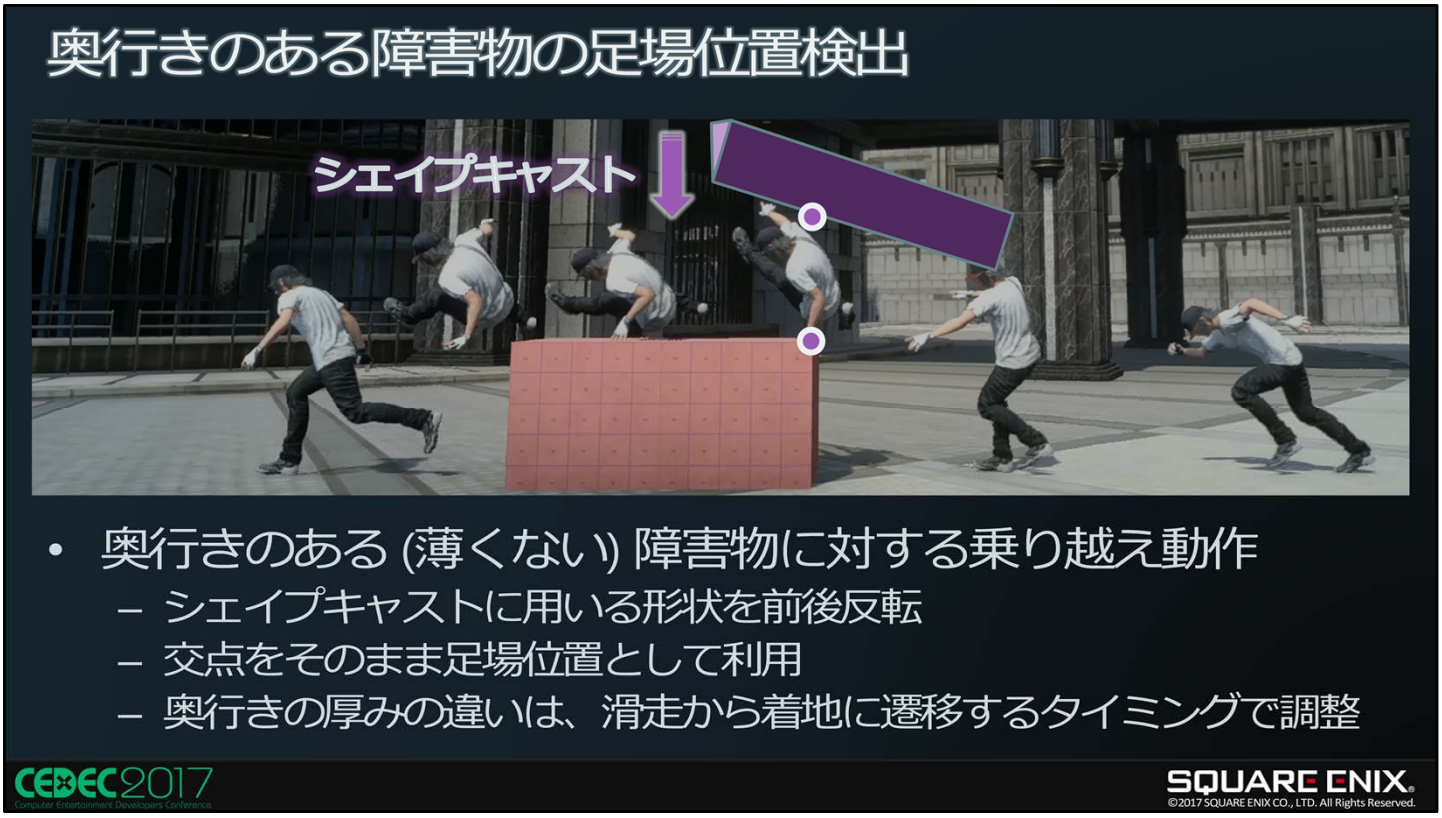
もし、厚みが大きすぎる場合には、乗り越え動作を実行するには不適当な足場と考えられます。




最後に、乗り越え動作を実行するために、足場の上の空間に余裕があることを確認します。

キャラクタの前方方向に行ったシェイプキャストが交点を持たなければ、乗り越え動作を実行できる空間の余裕があると考えられます。

必要な空間の余裕の大きさは、アニメーションクリップに対するデータとして予め定義しておきます。

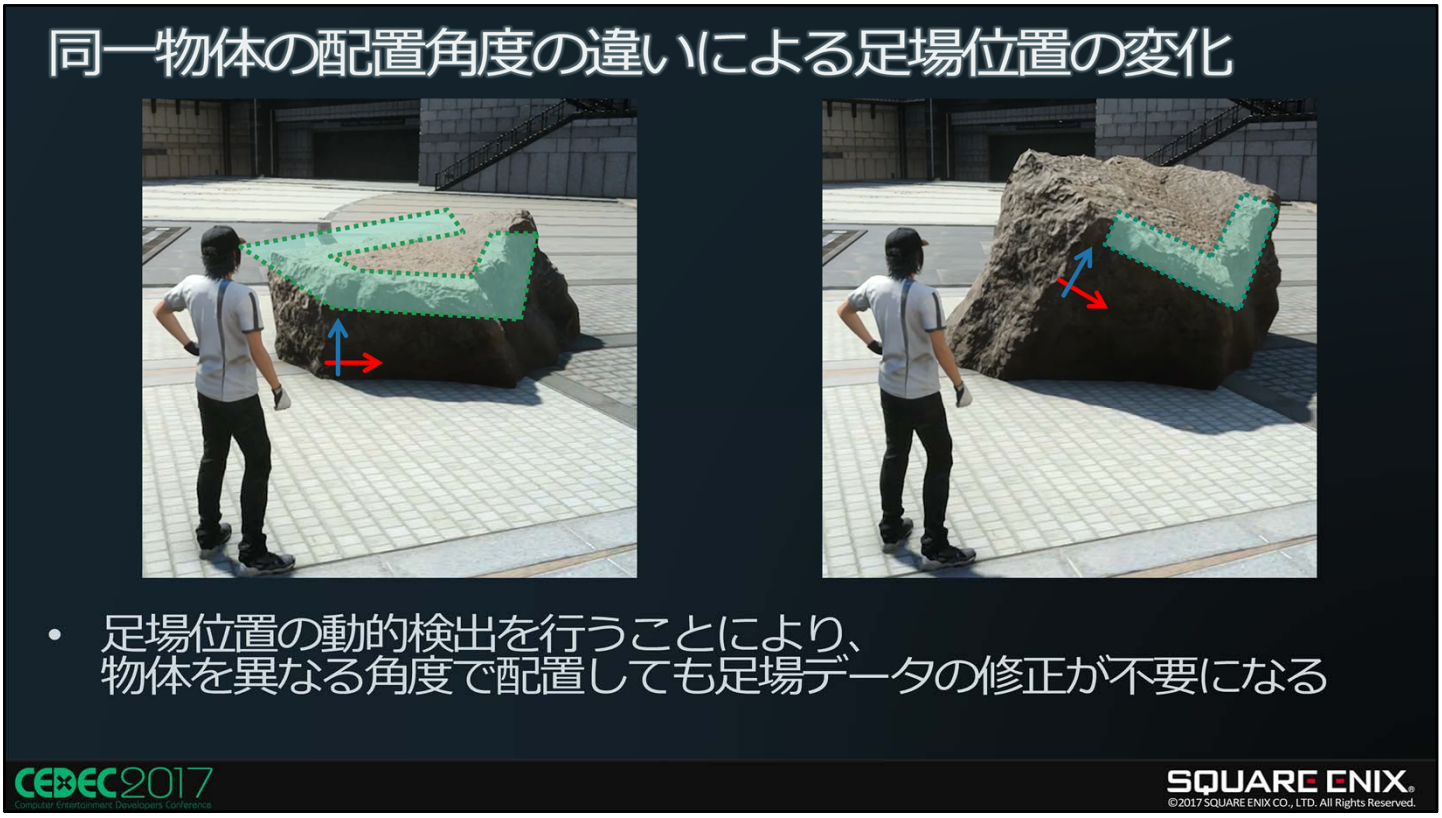



これまでのスライドでは、薄い壁のような障害物に対する足場位置の検出について説明してきました。

これとは異なる障害物の形状として、奥行きのある物体に対する乗り越え動作については、足場位置を検出するときのシェイプキャストの形状を前後反転したものを利用します。

このとき、障害物の奥行きに対する制約はなくなり、シェイプキャストの交点がそのまま足場の位置となります。

奥行きの違いは、乗り越え動作を実行するときに、障害物の上を滑る状態から着地動作に遷移するタイミングによって調整します。




実際のワールドマップでは、同じ形状の障害物を配置する場合でも、異なる角度で置かれることがあります。

このような場合でも、ゲーム実行時に動的な足場位置の検出を行うことで、キャラクタとの位置関係を考慮して、適切に足場の位置を計算することができます。

また、配置の角度が異なる場合でも、乗り越え動作に関するデータを変更する必要がありません。

## まとめ: 乗り越え可能な障害物の検出

- 障害物上の足場位置検出方法
  - a. 詳細な足場形状を静的に作成する方法
  - b. 足場位置を動的に検出する方法
  - → データ量・処理負荷に一長一短がある
- 動的な足場位置検出
  - – レイキャスト・シェイプキャストを組合せる
  - – 同一の障害物が異なる角度で配置されても、自動的に対応できる
  - – 足場が水平に近く、凹凸が少ないことを検査できる

CEDEC2017 Computer Entertainment Developers Conference

SQUARE ENIX

©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

障害物の検出についてまとめます。

足場の位置を検出する方法については2つに分けることができ、一つは詳細な足場の位置を予め静的なデータとして作成しておく方法、もう一つはゲームの実行時に動的に足場の位置を検出する方法です。

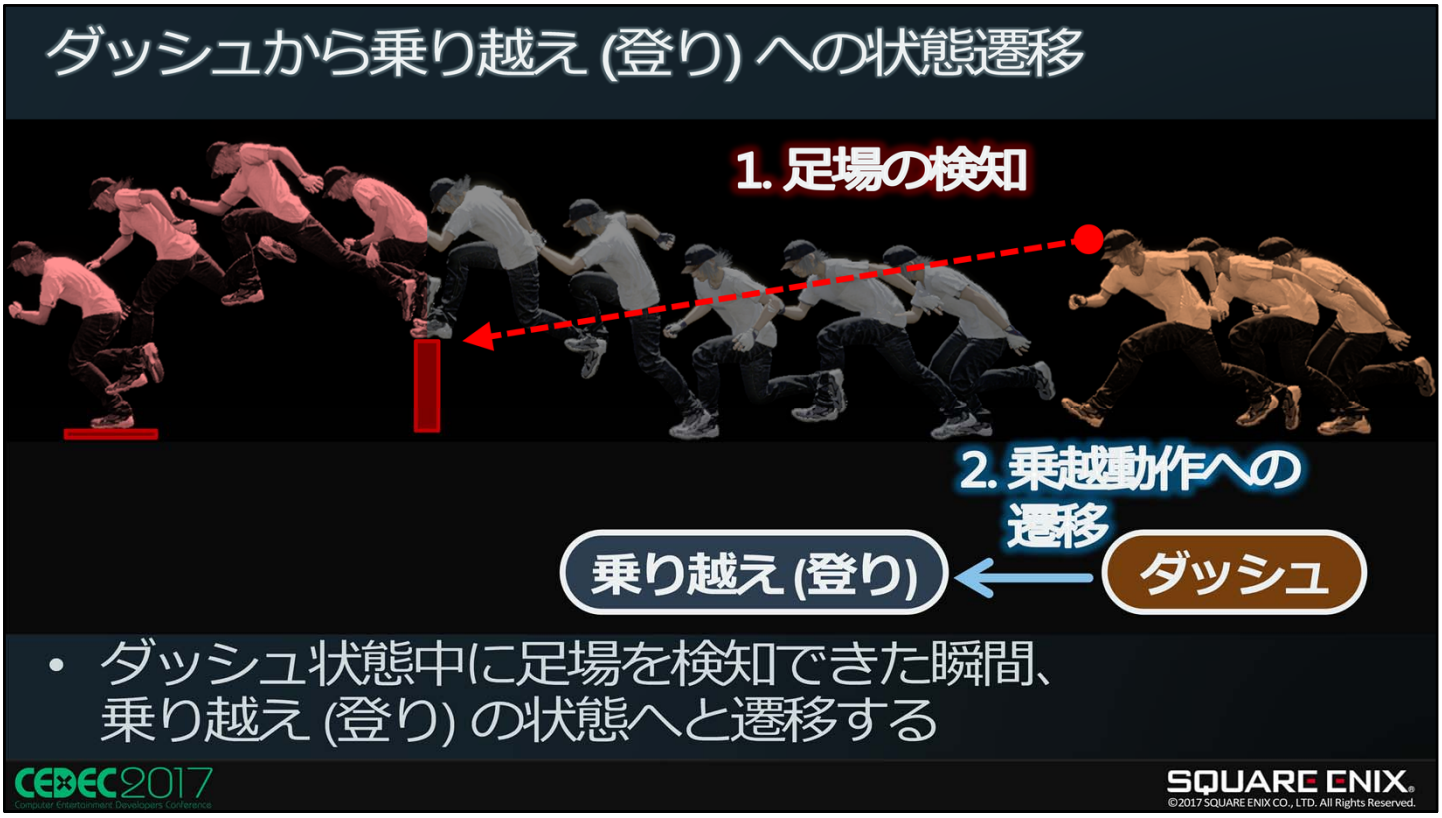
この2つの方法はデータの量や処理負荷に一長一短がありますが、FFXVでは後者の方法を利用しています。

後者の、動的に足場の位置を検出する方法では、レイキャストとシェイプキャストを組み合わせることで足場の位置を決定しています。このとき、シェイプキャストの形状を工夫することで、障害物の足場となる場所がある程度平らであるかどうかを検査しています。

次に、足場の位置を検出したときに、アニメーションを乗り越え状態に遷移させる仕組みについて説明します。

このスライドは、アニメーション3つの状態、ダッシュ、乗り越えの登り、乗り越えの着地をあらわしています。

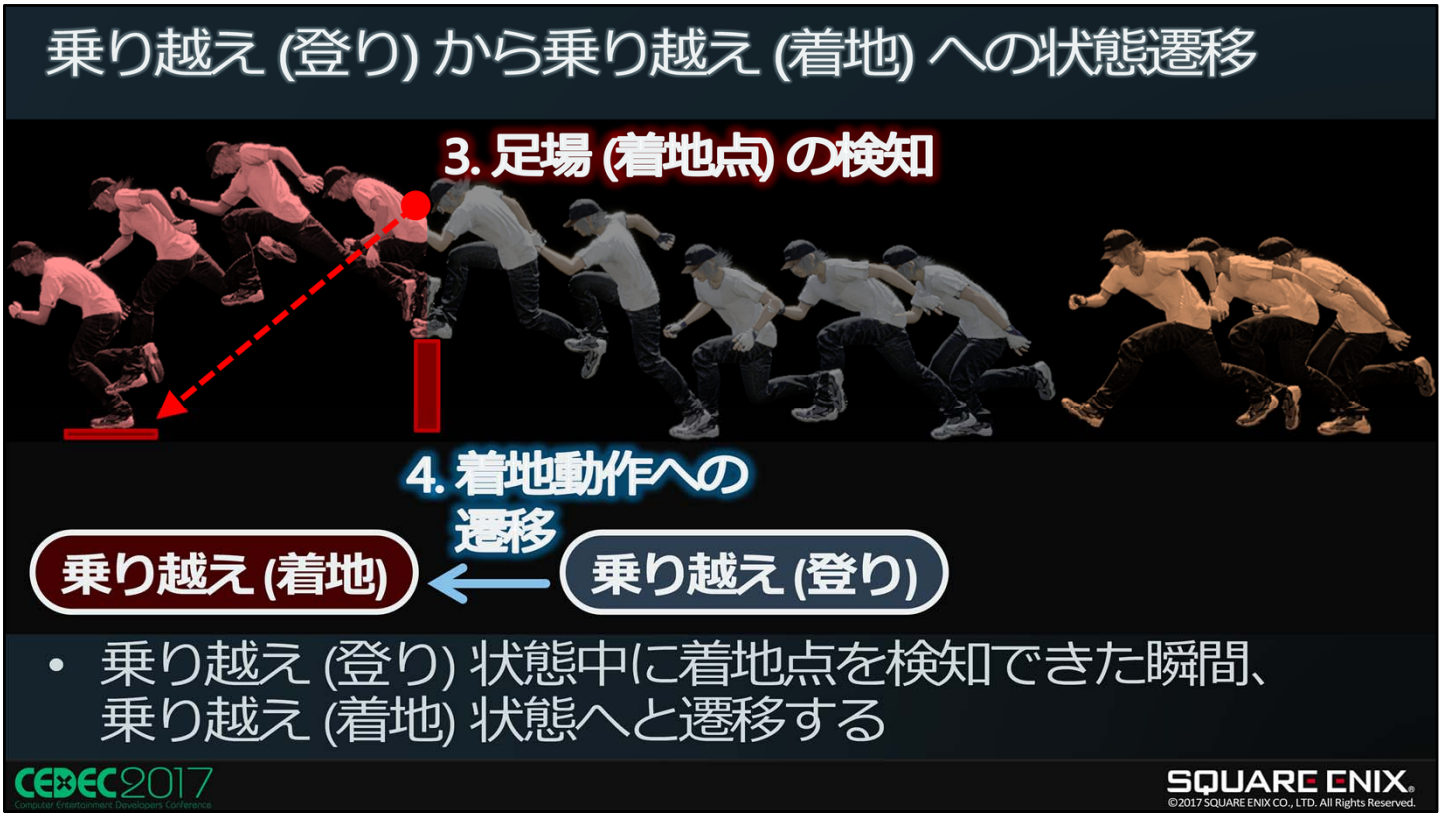
これまでの説明では乗り越え状態は一つの状態としていましたが、実装したシステムでは障害物に登る状態と、障害物から着地する状態を分けて取り扱っています。




アニメーションがダッシュ状態にあるときに、キャラクタの前方に乗り越えられそうな障害物を検知すると、そのタイミングで乗り越え状態に遷移します。

アニメーションシステムには、イベントが発行されるとアニメーションの状態を遷移させる仕組みがあるので、その仕組みを利用します。

つまり、ダッシュ状態中に障害物が発見されたら、状態遷移を行わせるイベントを発行するようになっています。

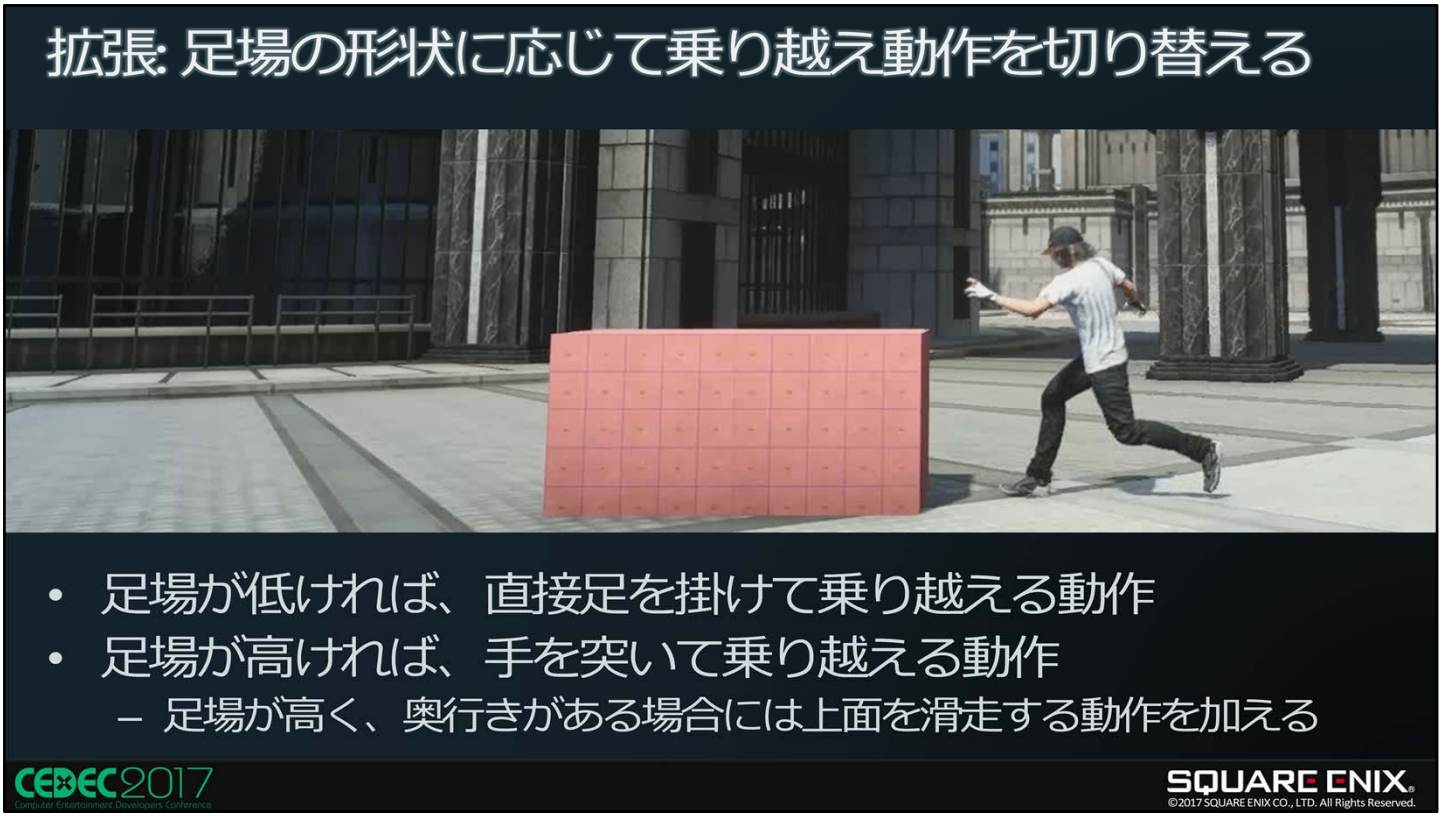



同様にして、乗り越えの登り状態にあるとき、着地できそうな地面が検知できたらイベントを発行します。

このイベントの発行によって、乗り越えの登りの状態から、乗り越えの着地の状態へと遷移します。

全体としては、乗り越え動作の開始から終了までは、スライドに示すような状態遷移のサイクルになります。

はじめに右端のダッシュ状態にあり、足場や着地点を検知次第、アニメーションの状態が次々に左の状態へと遷移していく仕組みです。

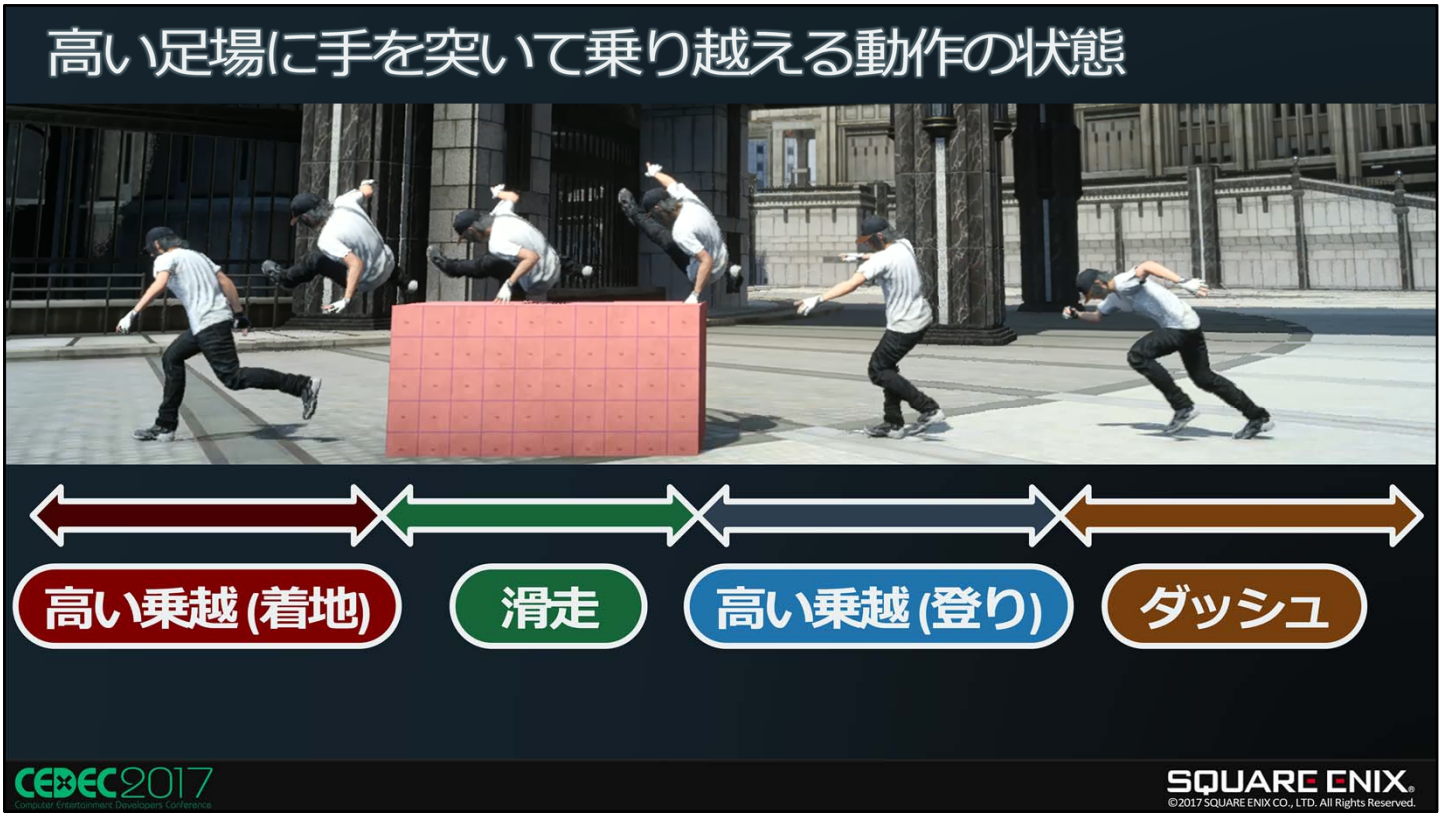



プレイヤーの乗り越え動作は一種類だけではなく、障害物の高さや形状に応じて異なる動作を選択して出し分ける仕様になっています。

足場が低いときには直接足をかけて乗り越える動作になります。

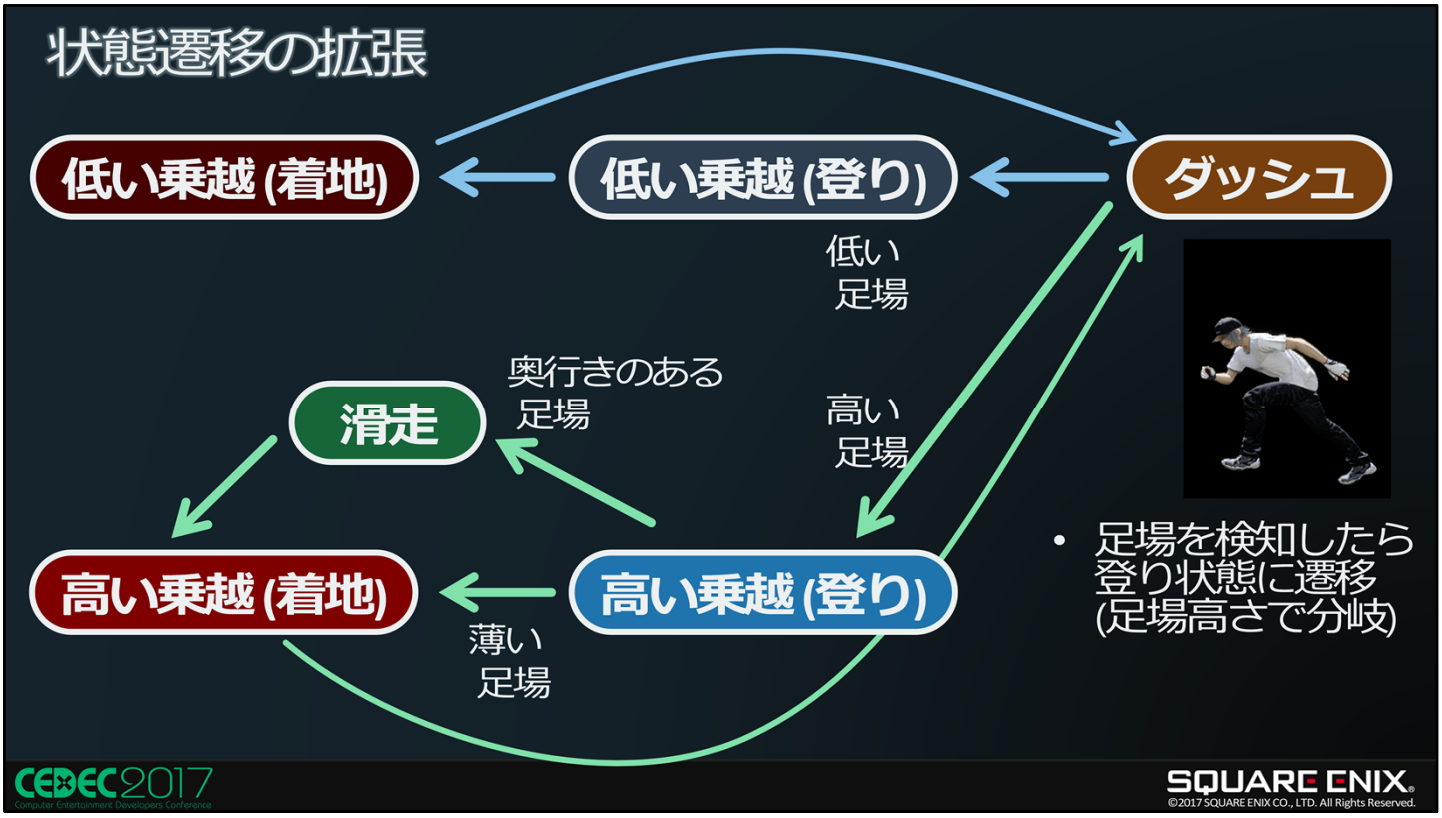
一方で、足場が高いときには、手を使って乗り越える動作になります。

足場が高くて、かつ奥行きがある障害物に対しては、障害物の上の面を滑るような乗り越え動作を加えます。




このような高い足場に対する乗り越え動作は、3つの状態を持ちます。

はじめに乗り越え動作の登る状態、つぎに障害物の上の面を滑る状態があります。最後に、足場から降りて着地する状態が続きます。



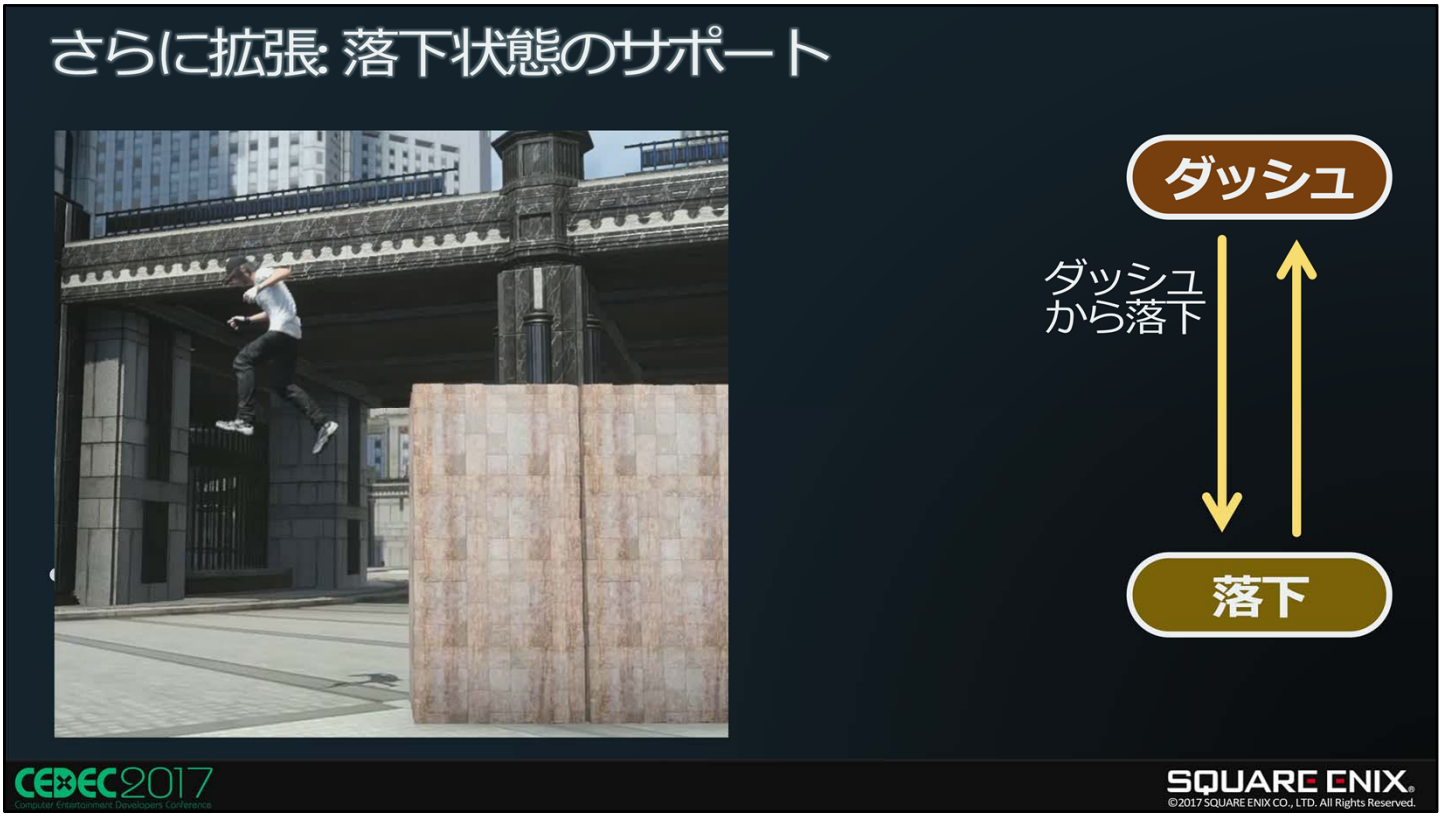

高い足場に対する乗り越え動作を統合したアニメーションの状態遷移は、スライドのようになります。

まず右端のダッシュ状態からはじまって、障害物を検知したときに、足場の高さが低いか高いかに応じて状態遷移を分岐させます。

高い足場を検知した場合は、下の分岐に遷移することになります。

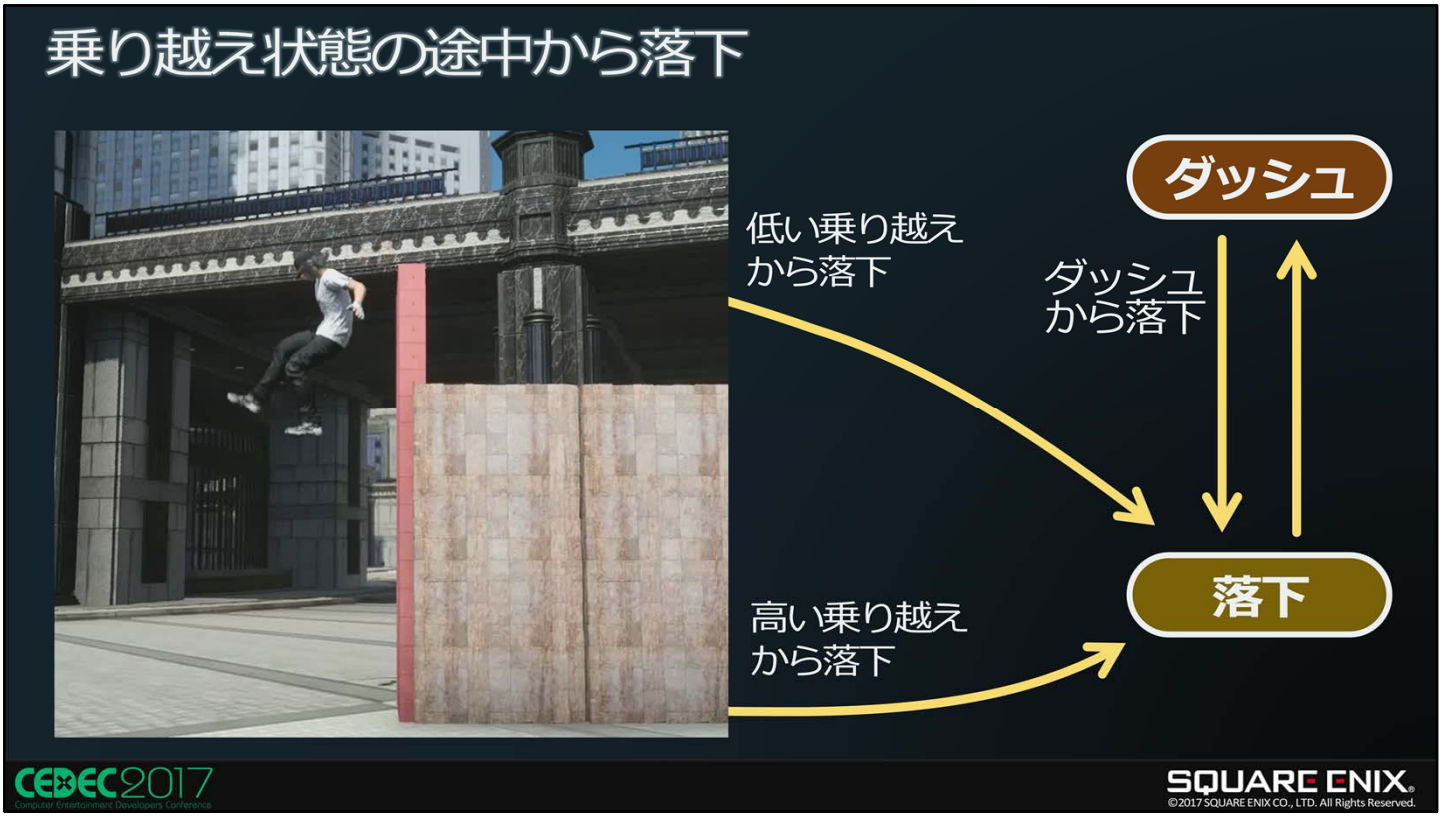
このとき、さらに足場の厚み・奥行きによって、足場の上の面を滑っていくか、そのまま着地するかという分岐を行います。

最終的には、着地状態を終えたらダッシュ状態に戻ることになります。



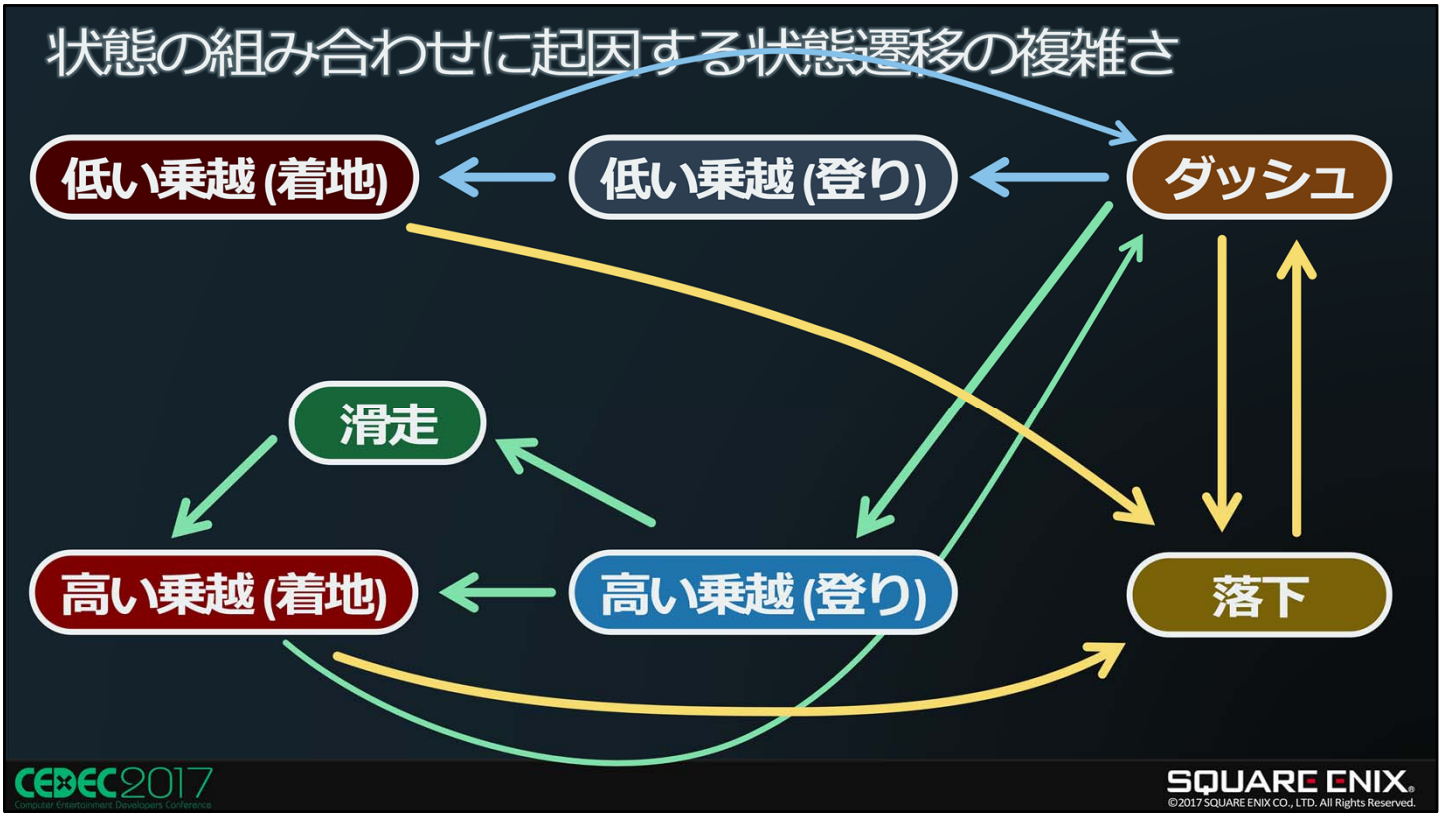

実際のゲームプレイでは、プレイヤーは平面だけを移動するわけではなく、高いところから落下する場合があります。

このような落下の動きは、アニメーションの状態としてはダッシュ状態からの遷移として定義できます。




同様に、障害物を乗り越えている途中から、乗り越え動作をキャンセルして落下する場合があります。

障害物の高さに対応して乗り越え動作を分岐させているので、低い乗り越え・高い乗り越えのそれぞれの状態の途中から、落下状態に遷移することになります。

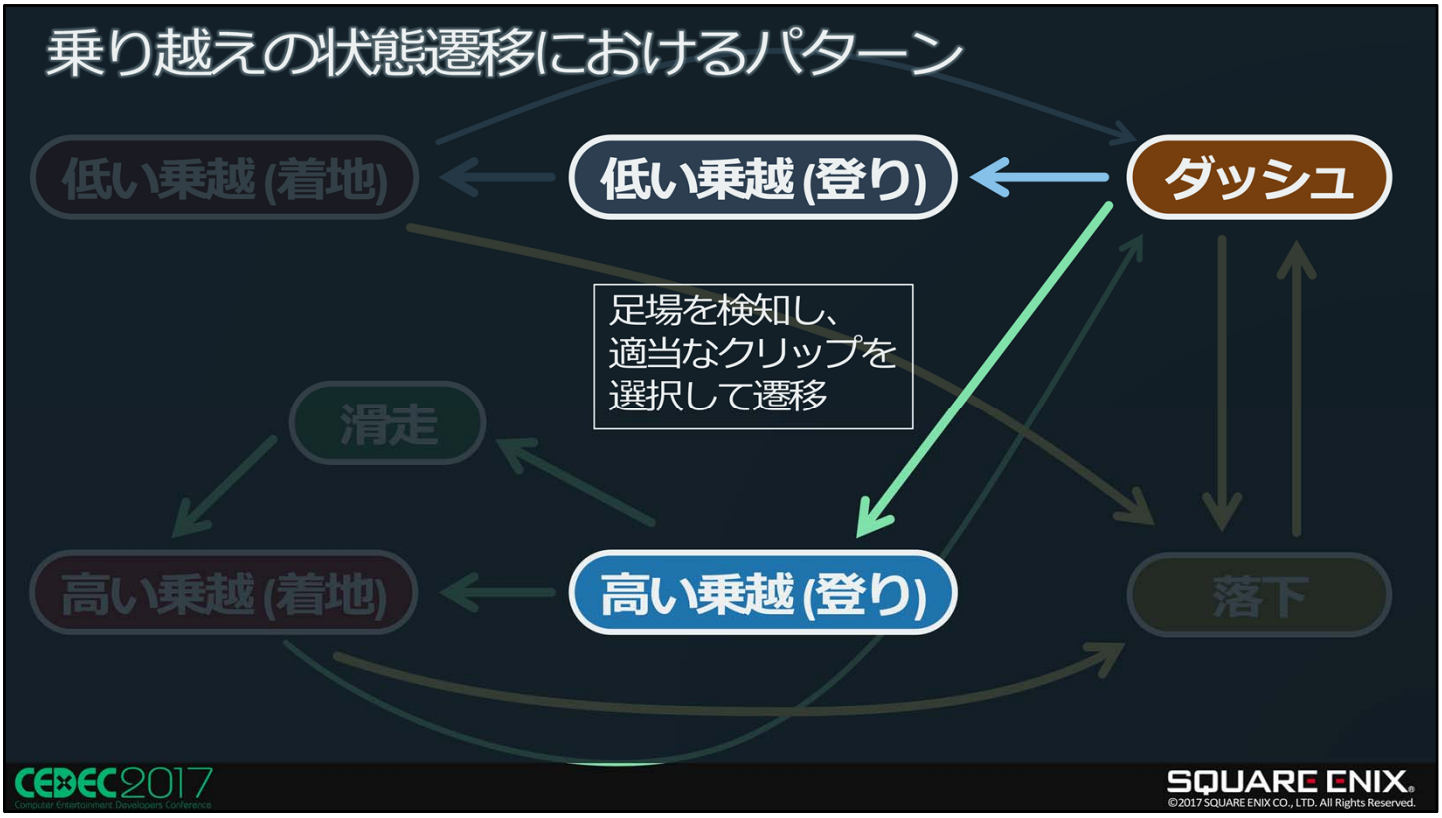



アニメーションに落下の状態を加えた状態遷移の全体は図のようになります。

落下がなかった場合に比べると、落下への遷移が増えた分だけ複雑になっています。

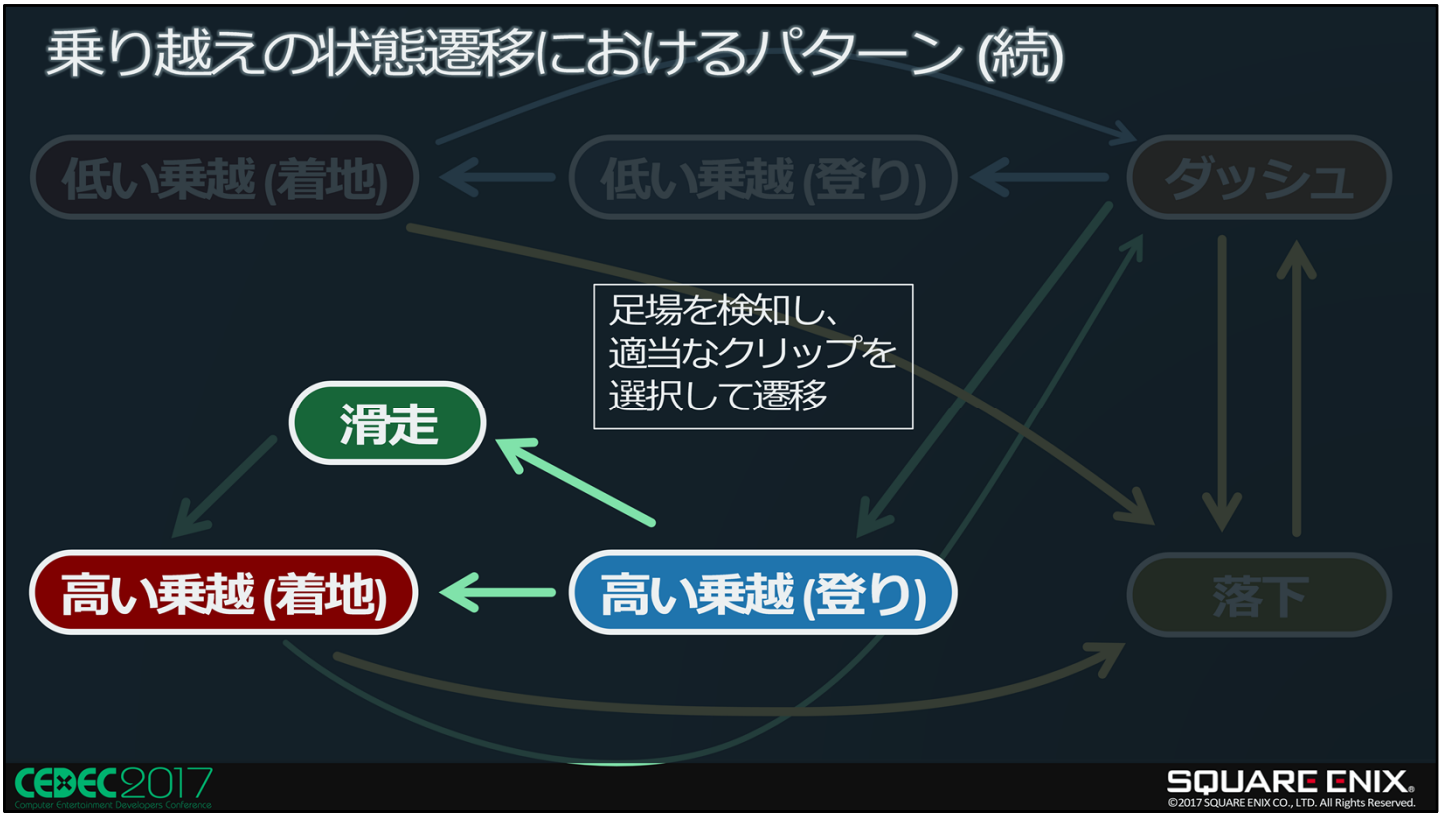
ここで問題となってくるのは、落下状態のような状態を別に増やしたり、乗り越え動作のパターンを増やした時に、考慮しなくてはならない遷移の数が状態の組み合わせによって増加し、状態遷移全体のメンテナンスがどんどん難しくなってしまうということです。

そこで、状態遷移をどうにかして、よりシンプルな形にできないかを考えます。

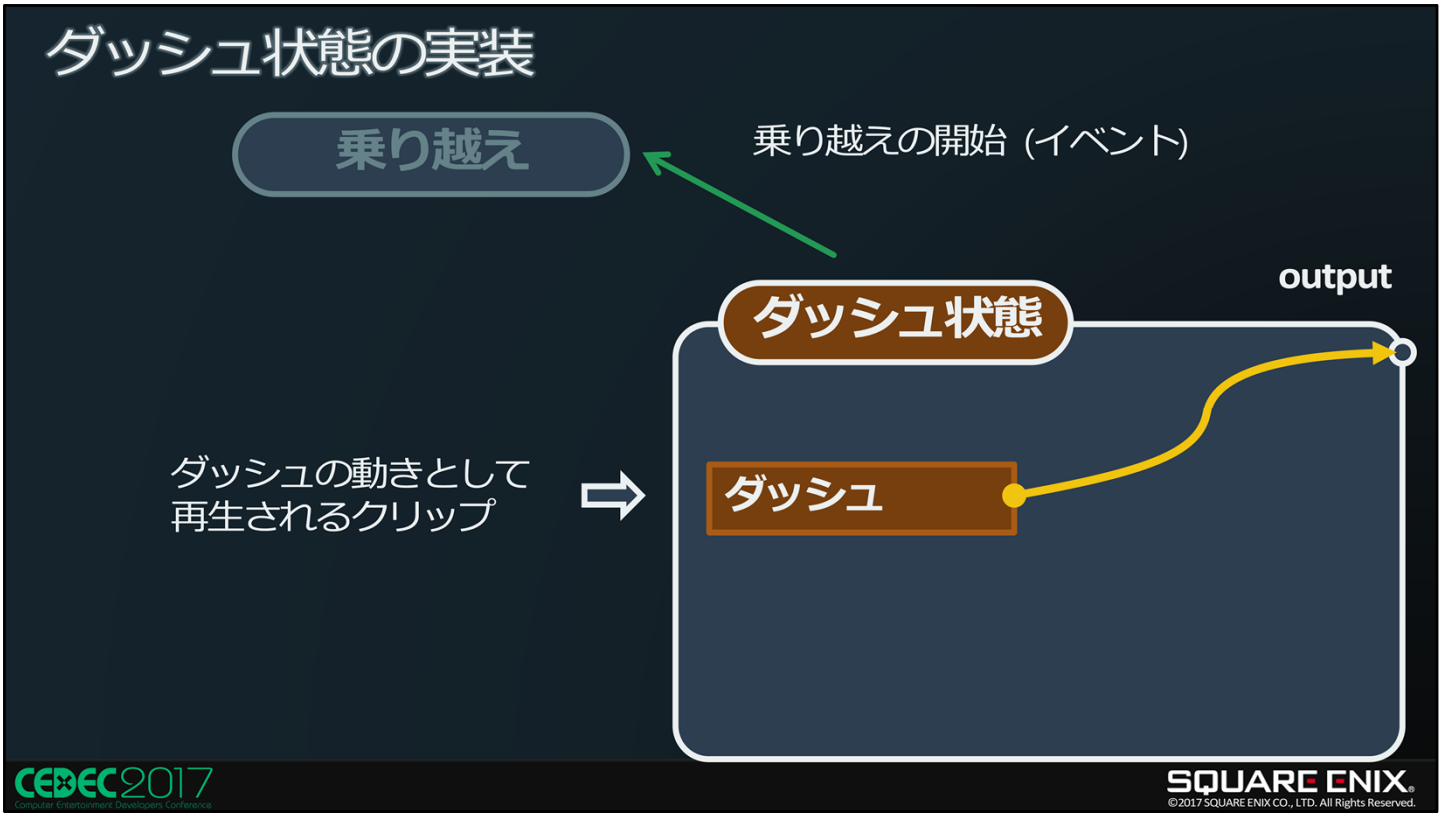



乗り越えの状態遷移には一定のパターンがあります。

障害物に足場を検知したら、適当な乗り越え動作のクリップを選択して遷移する、というパターンが見えます。



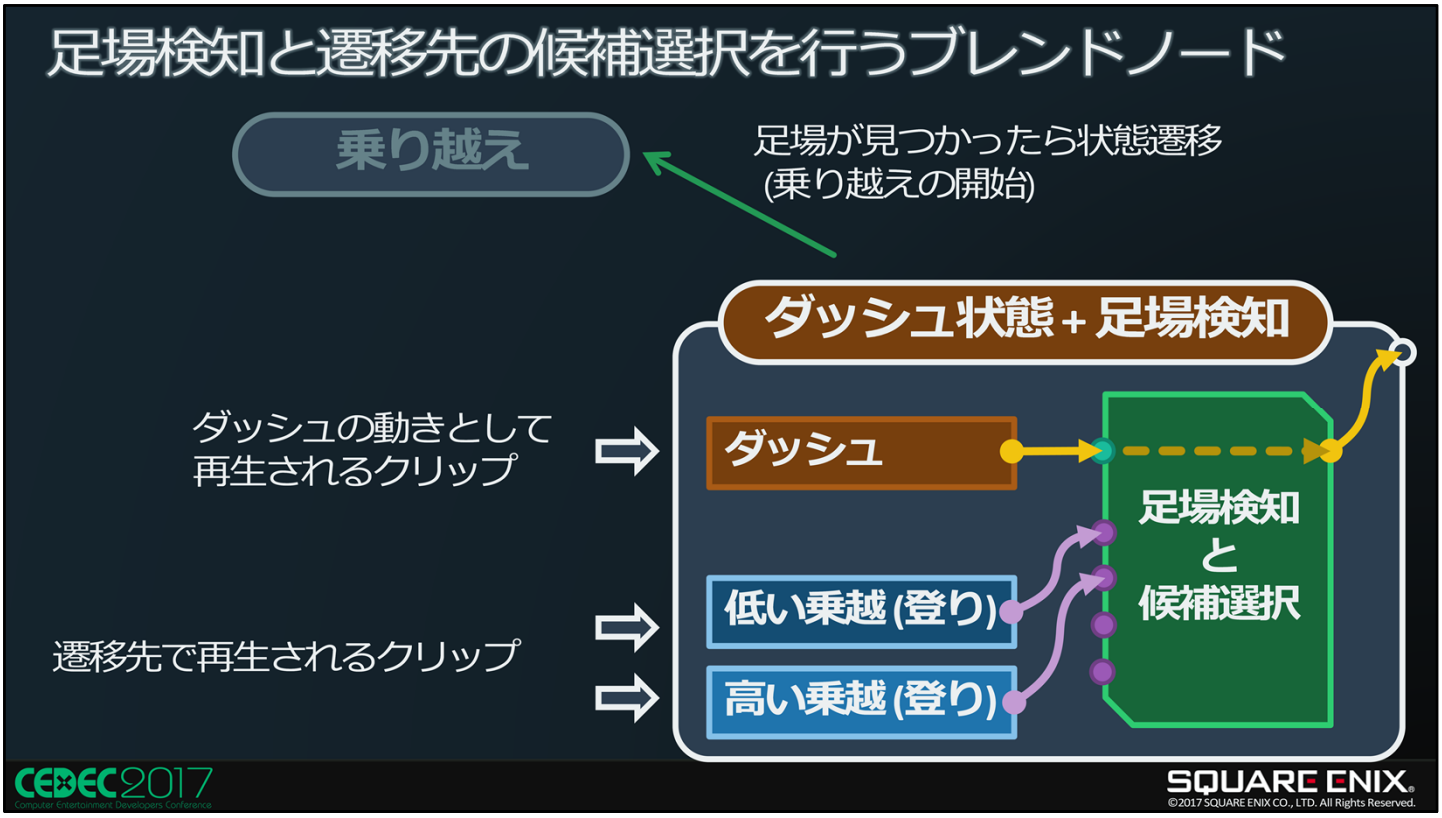

高い足場に対する乗り越え動作では、上の面を滑るか、そのまま着地するかを選択して遷移という形になります。




ここで、足場を検知して遷移するという機能をアニメーションの状態としてどのように実装するかという点について説明します。

スライドは、ダッシュ状態をダッシュのアニメーションクリップを再生することで動きを定義している様子をあらわしています。

このとき、適当なタイミングで乗り越え動作を開始するというイベントを通知することで、ダッシュ状態から乗り越え状態に遷移させます。



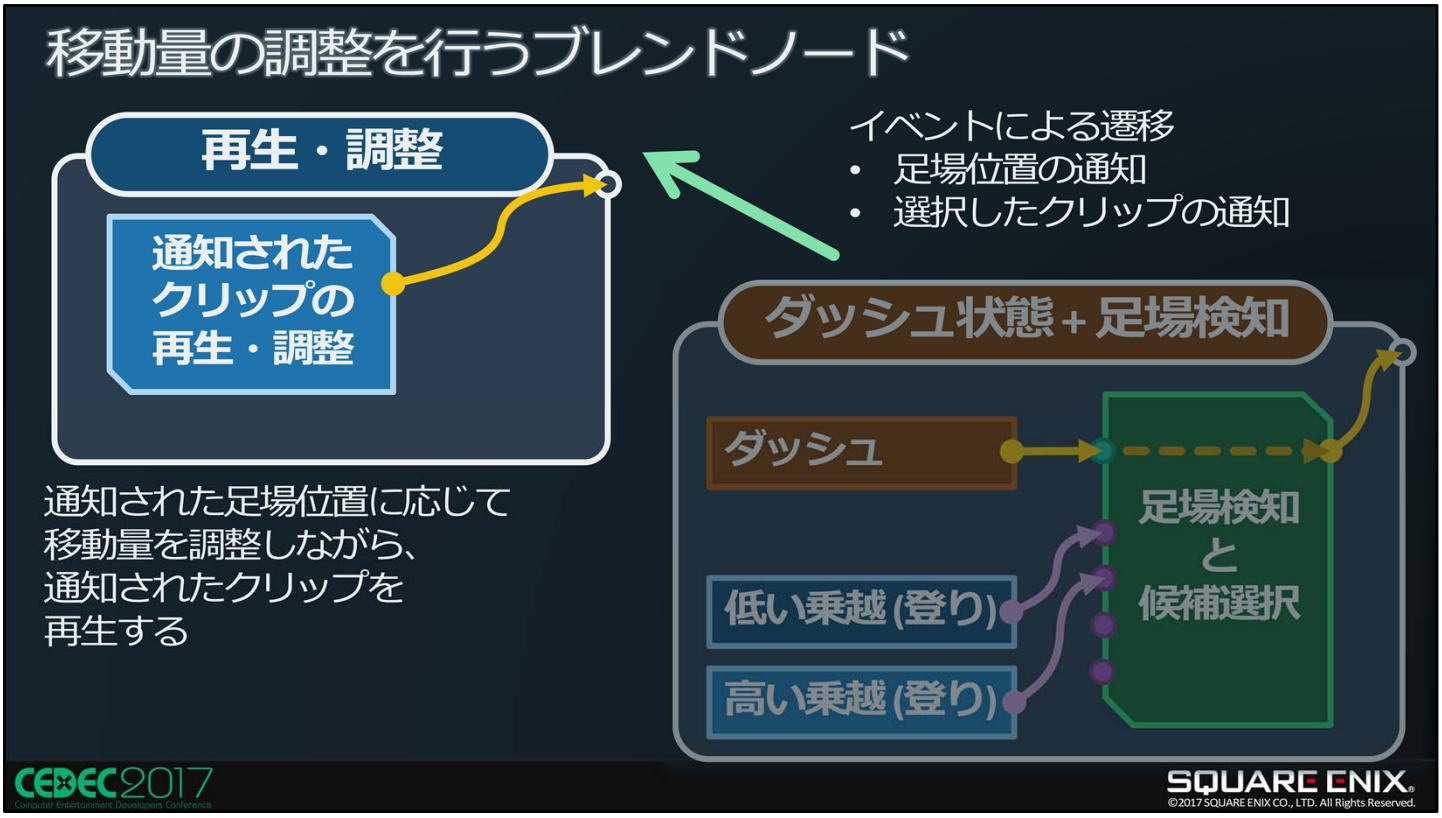

つまり、ダッシュ状態にあるときには、ダッシュの動きを再生すると同時に、障害物の足場を検知する必要があります。

本発表の実装では、新しいブレンドノードとして、足場の検知を行うブレンドノードを導入しました。

この足場検知のブレンドノードは、ダッシュのアニメーションクリップが再生されているときだけアクティブになって、キャラクタの進行方向の障害物を検知します。

ただし、アニメーショングラフ上での機能はブレンドノードですが、入力されるダッシュの動きには手をくわえず、そのままの形で出力に流します。

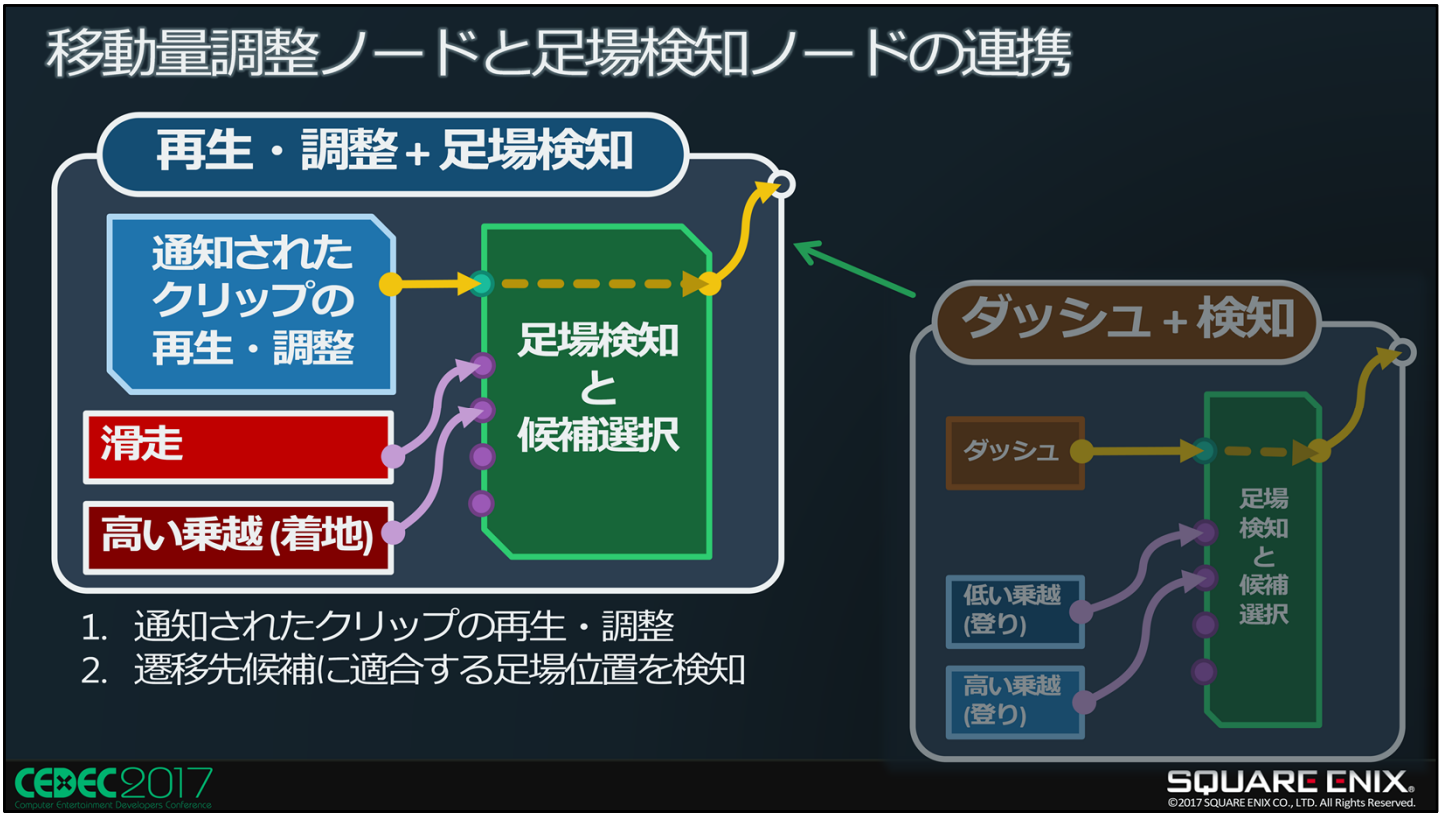
また、このブレンドノードにはもう一系統の入力があって、この入力には遷移先の乗り越え状態で再生するクリップを設定しておきます。クリップは複数設定できるので、障害物を検知したとき、その配置などに応じて、このクリップの中から適当なものを選びます。




検出した足場の情報は、遷移を起こすイベントの一部として遷移先に通知します。

また、足場の配置や大きさに対応して、候補の中から選択された乗り越え動作のクリップも同時に通知されます。

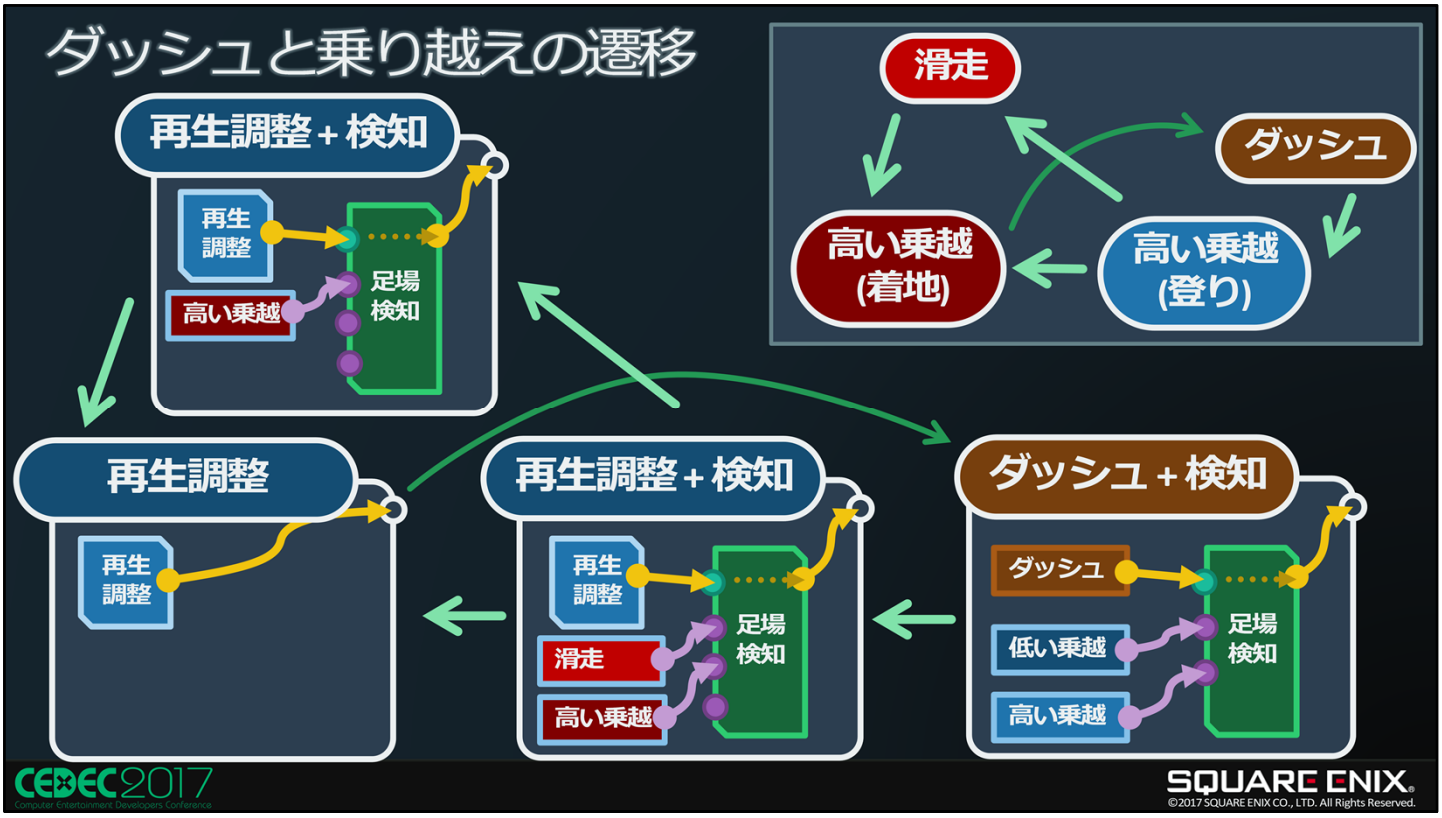
遷移先の状態ではこの通知を受けて、足場の位置に応じて移動量を調節しながら、選択されたクリップを再生することになります。そのために、専用のブレンドノードを用意してあります。




高い足場に対する乗り越え動作で足場の上を登っているときは、その次の状態は障害物の上を滑るか、そのまま着地するかのどちらかへと遷移を分岐させる必要があります。

どちらに分岐するかは検出される足場の位置に依存しているので、足場を登る動作を再生する状態でも、ダッシュ状態と同じように足場を検知するブレンドノードを追加します。

また、再生中のクリップから遷移可能な動作を表すアニメーションクリップ (滑走と高い乗り越え) を、遷移先で再生する動作として入力に追加しておきます。

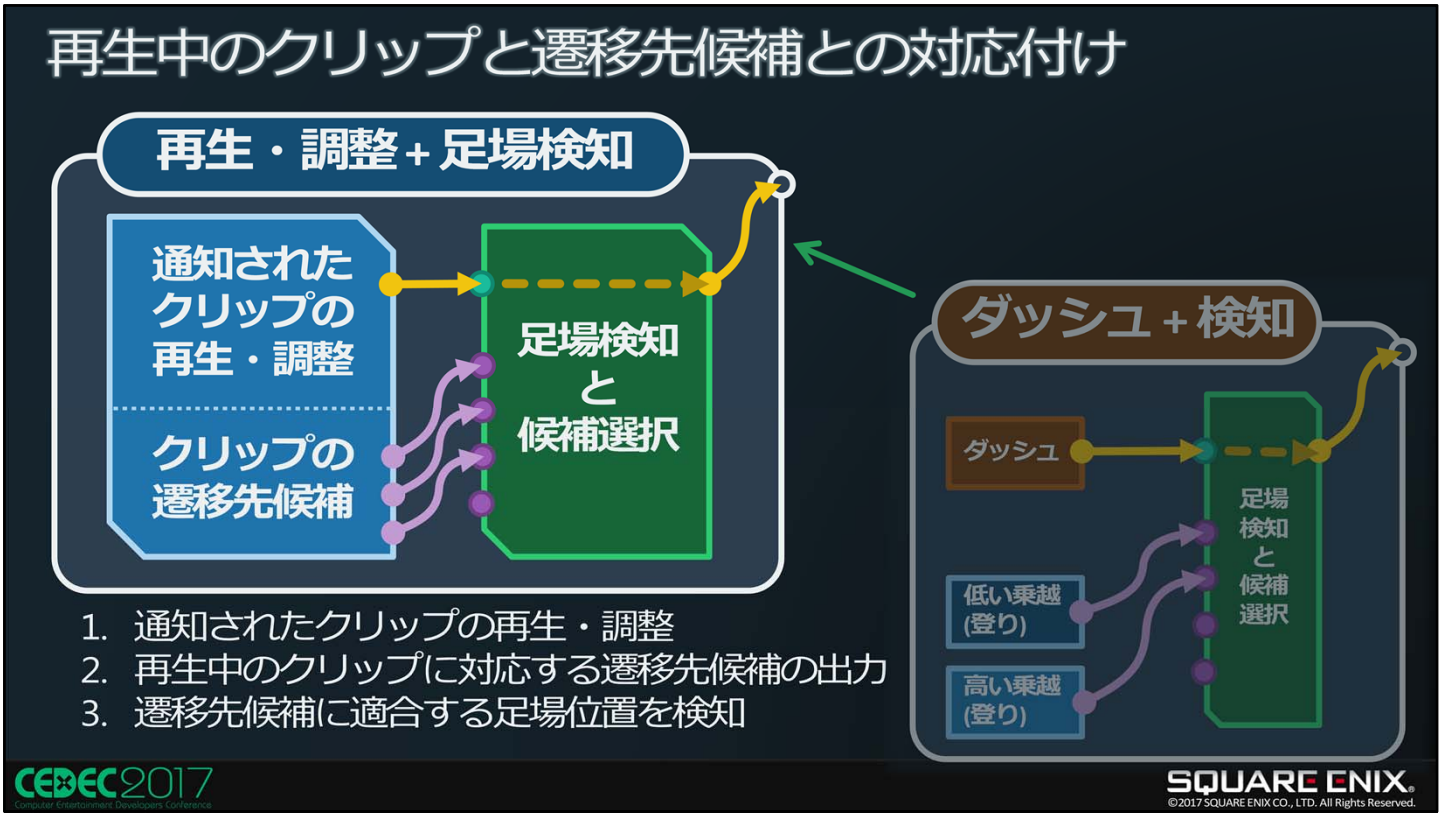



このように、足場を検知するブレンドノードと、動作を再生するブレンドノードを連携させることで、足場を検知して次々に状態を遷移する動作を実現することができます。

高い足場に対する乗り越え動作はスライドの右上のような状態の遷移として表すことができましたが、これをブレンドノードを連携させて実装したものが、スライドのようなアニメーショングラフになります。

それぞれの状態には、次に再生する候補となるアニメーションクリップがブレンドノードの入力として設定されています。

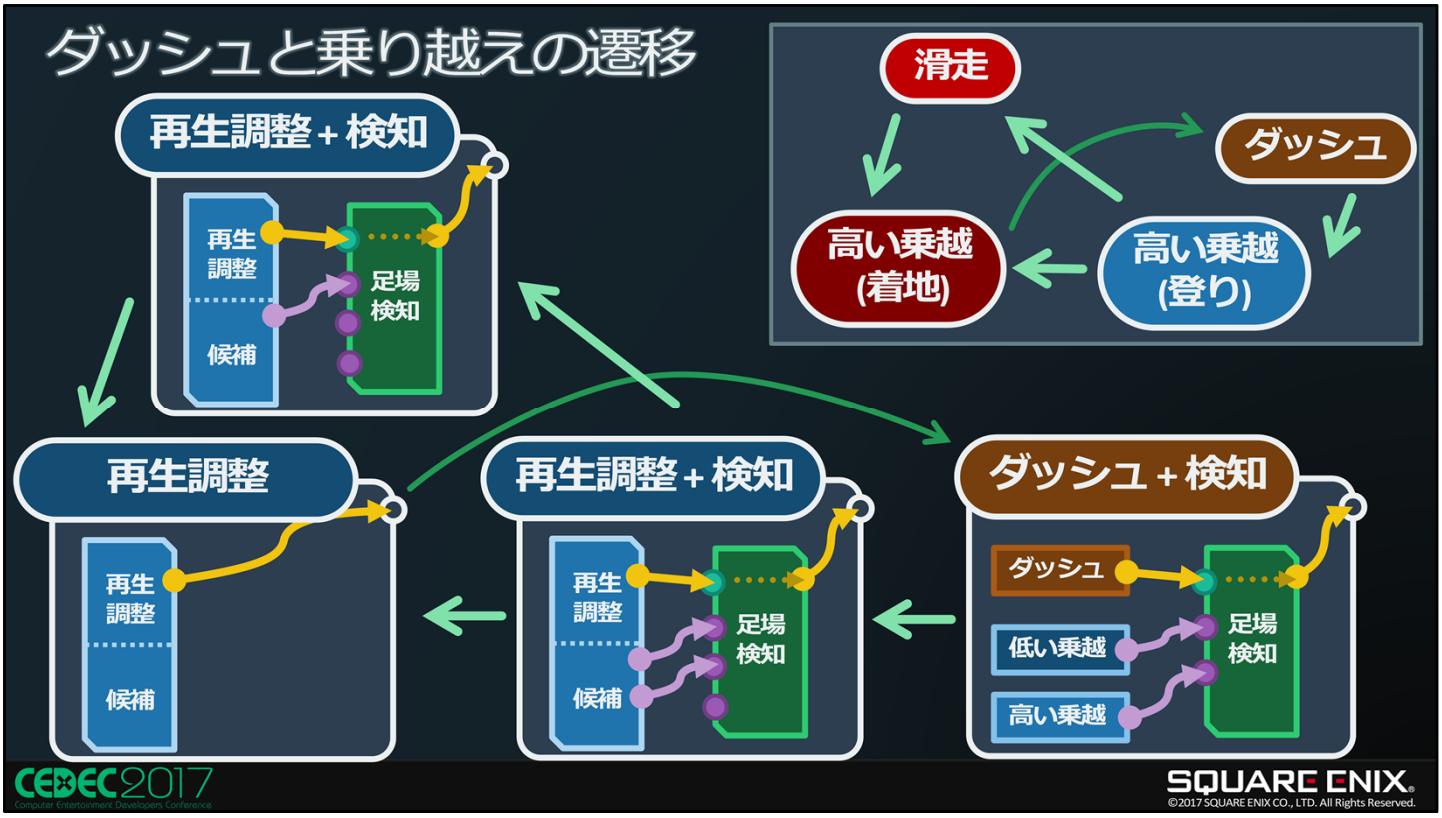
ここで一つ気がつくのは、再生中のアニメーションクリップがわかれば、次に再生する候補となるクリップは、事前に対応付けしておくことができるということです。




そこで、クリップを再生するブレンドノードを拡張して、再生中のクリップから、次に遷移できるクリップの候補を出力する機能を加えます。

この対応付けは、各クリップに対するヒントとして予めデータを与えておきます。

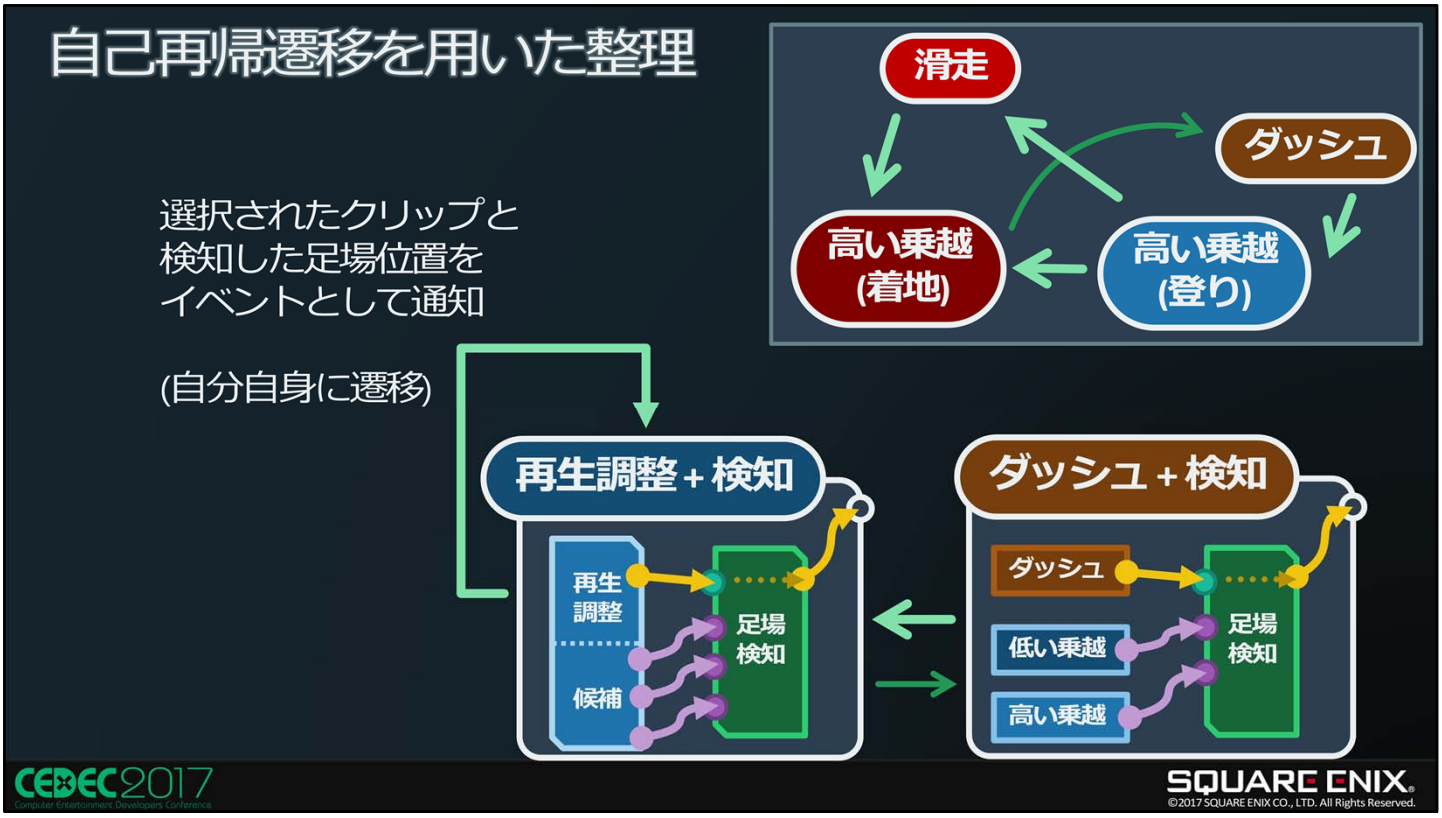
このようなブレンドノードの拡張によって、足場の検知を行うブレンドノードに対しては、直接クリップを指定するのではなく、再生中のクリップから対応づけられる候補を遷移先として入力することができます。




拡張した再生ブレンドノードを用いると、高い乗り越え動作に関する状態遷移はスライドのように実装することができます。

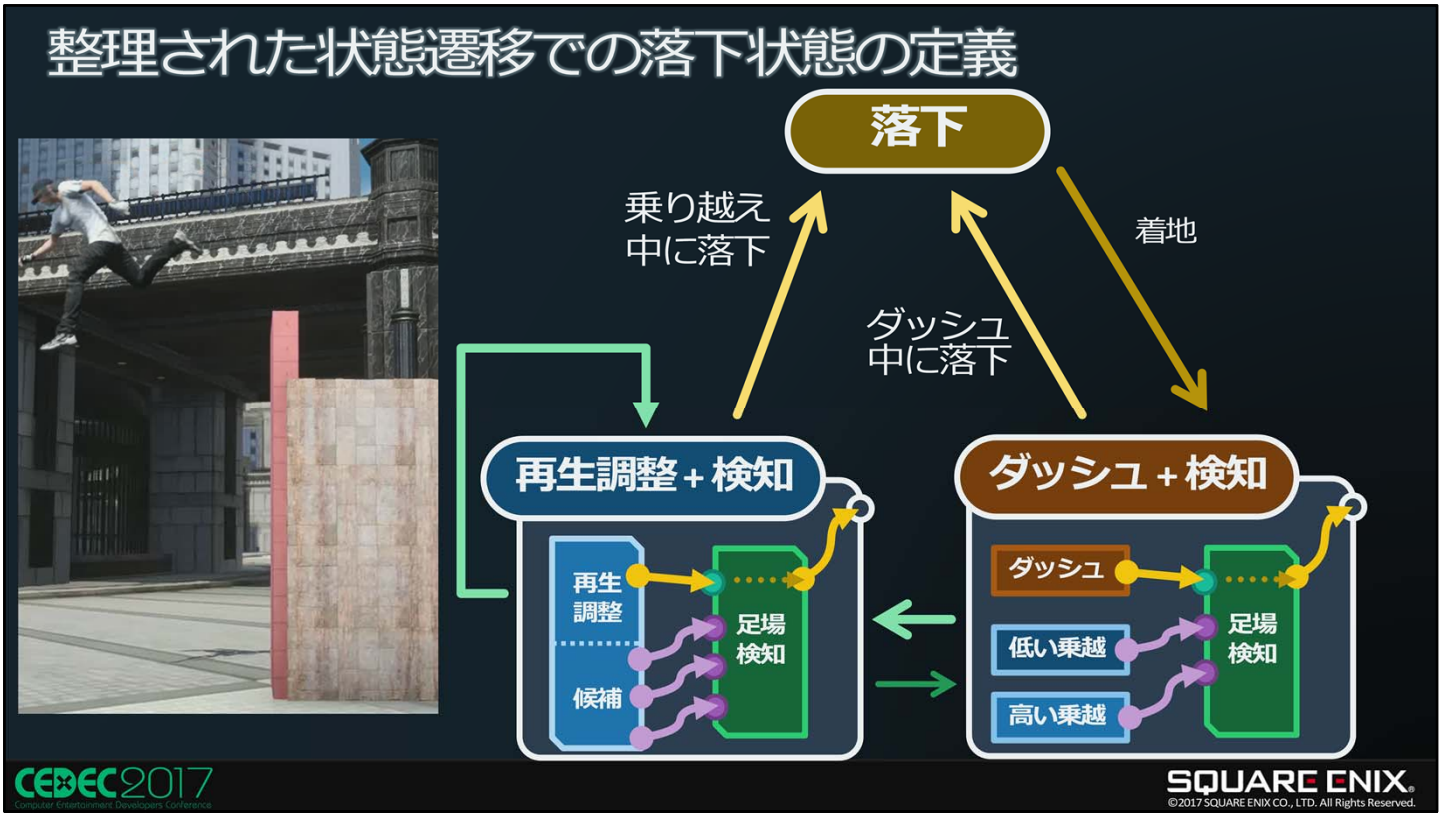
この状態遷移では、左の3つの状態では遷移先の候補となるクリップは自動的に対応付けられるので、直接指定しなくてよくなっていることがわかります。

またこのとき、左の3つの状態は、ほぼ同じ構造の状態になっていて、一箇所にまとめられそうに見えます。



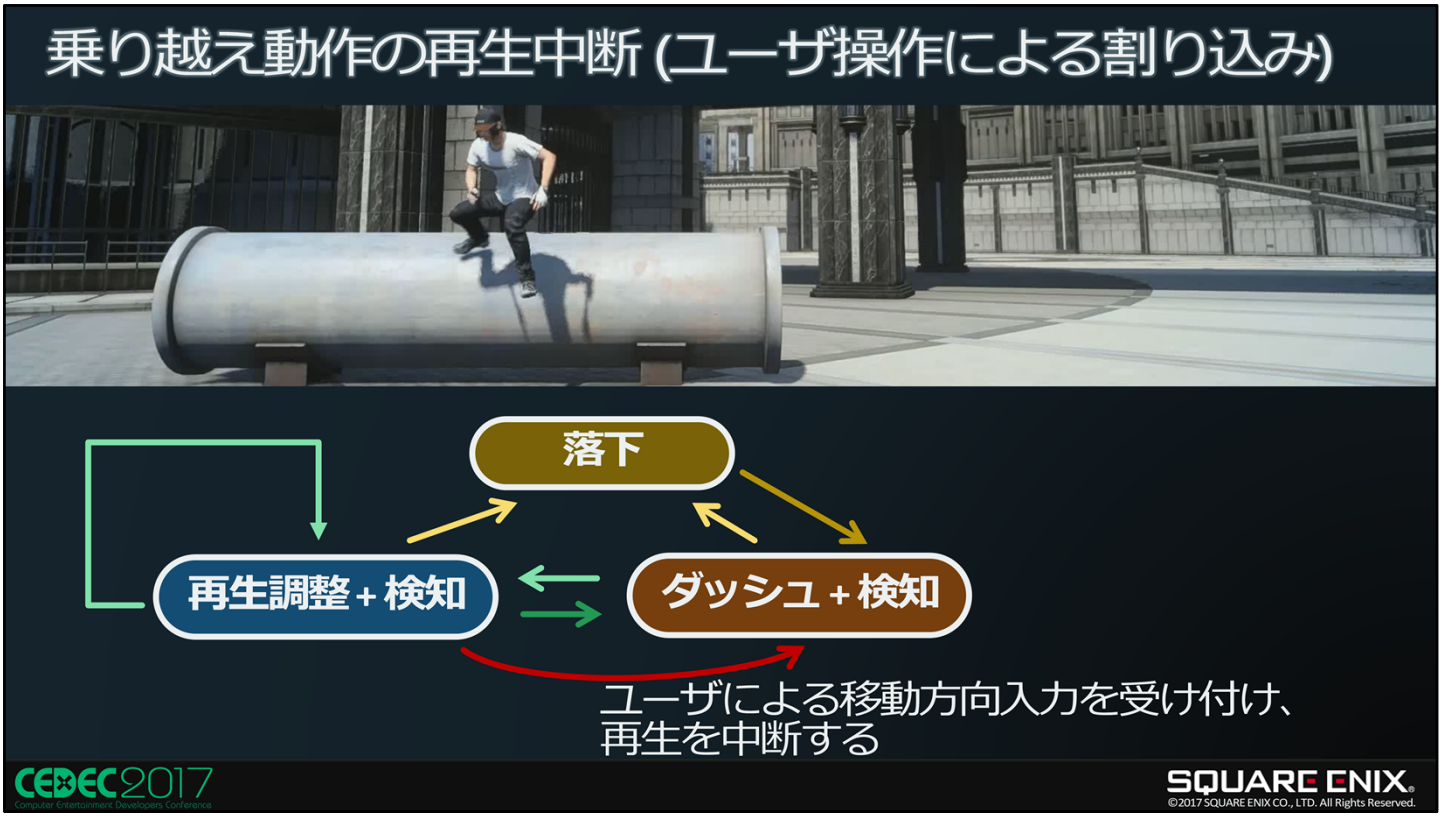

これらの状態を一箇所にまとめるためには、自分自身に遷移する構造を使います。

このような再帰的な遷移を導入することで、分岐を含む複雑な状態遷移を一箇所に集約することができます。




状態遷移をこのように一箇所に整理することで、落下状態を含む状態遷移もシンプルな形に整理できます。

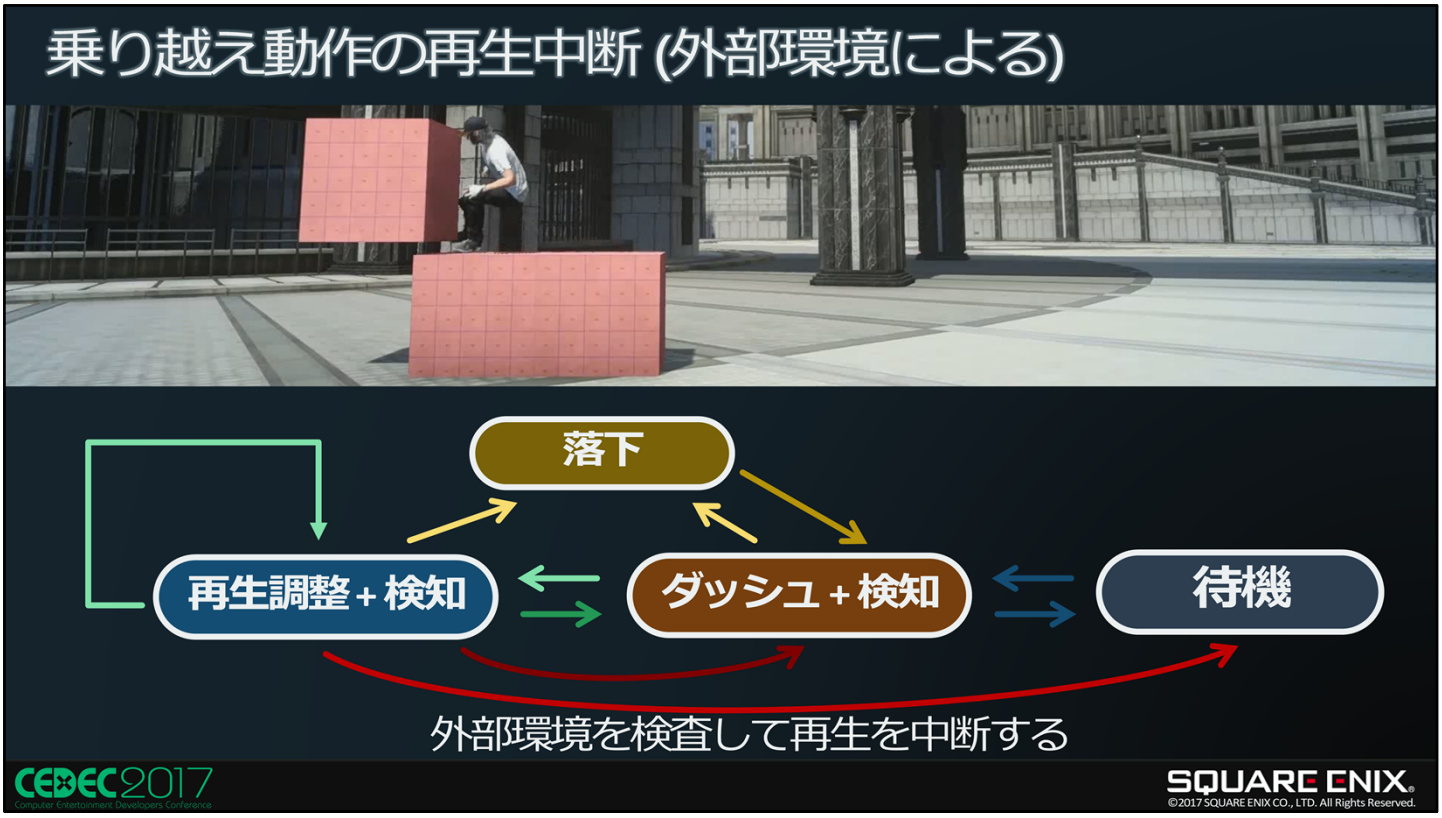
いままでは一つ一つの乗り越えを表す状態から落下することを考慮する必要がありましたが、整理した状態遷移では、乗り越え状態を表す一箇所の状態から落下する遷移だけを考えればよいことになります。




また、落下以外の遷移も簡単に追加することができます。

乗り越え動作を再生しているとき、途中で動作を中断したいときがあります。スライドでは、ユーザが移動方向を入力することで、障害物の上を滑る動作を中断する様子を表しています。

このような途中でキャンセルするような遷移を追加しても、乗り越え動作の再生を表す状態を一箇所に集約しておくことで、グラフに対する変更を最小限にすることができます。




外部の環境によって動作を中断しなければならない場合にも、同様に簡単な変更で対応できます。

この場合は、障害物の上を滑っているときに何かにぶつかったり、滑らない性質の障害物に差し掛かった時に再生を中断し、待機状態に戻るという遷移を追加しています。

## まとめ: 「乗り越え状態」への遷移と離脱

- 状態遷移をサポートするブレンドノード
  1. 乗り越え動作の選択と足場位置の検出
  2. 移動量の調整と遷移先クリップ候補群の保持
- アニメーションの一状態としての乗り越え状態
  - 落下やユーザ割込みのような中断処理をシンプルに記述できる
  - アニメーションデータ記述の一貫性を保てる

CEDEC2017 Computer Entertainment Developers Conference
SQUARE ENIX ©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

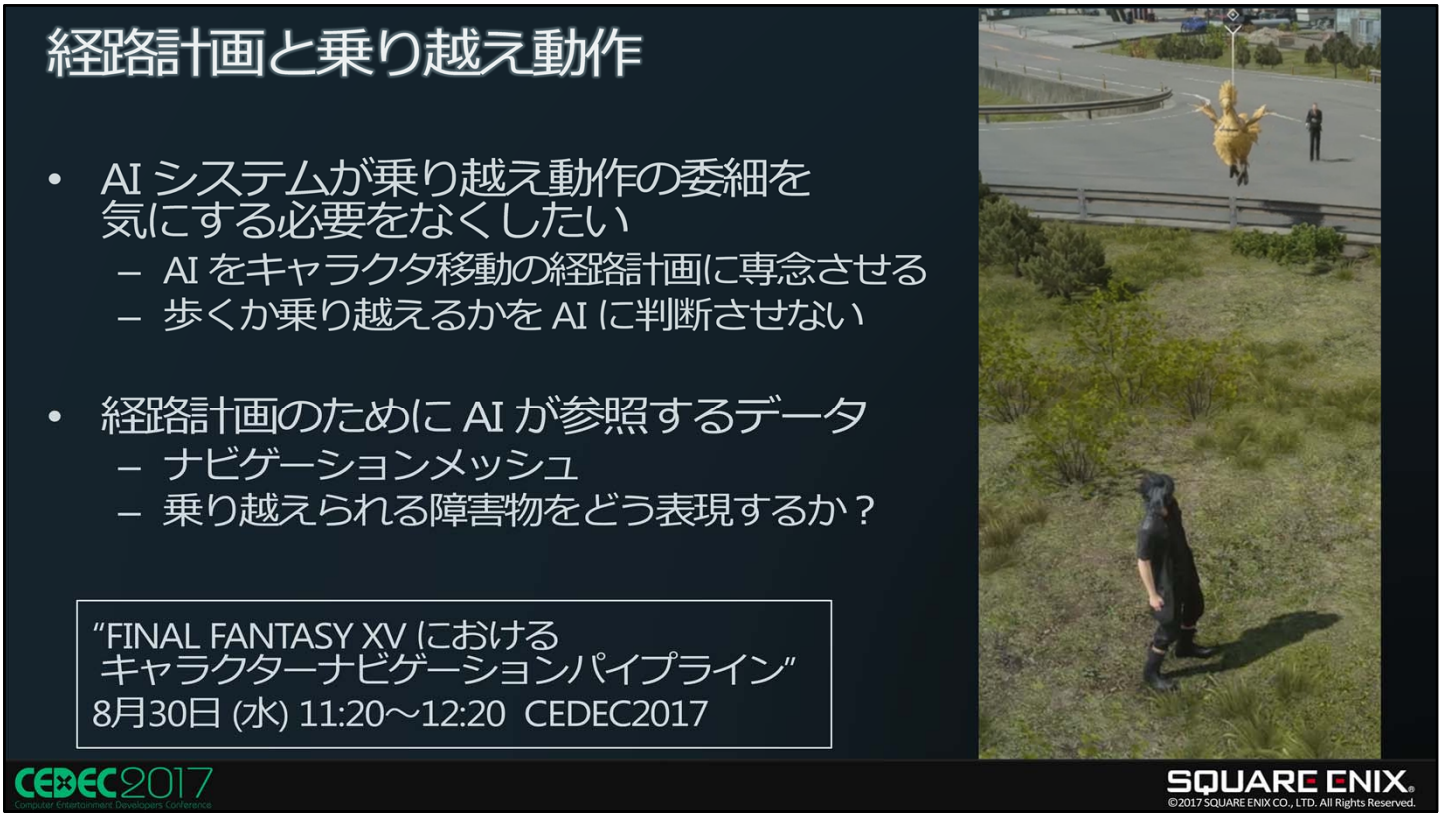
このように、乗り越え動作の状態遷移に必要な2種類のブレンドノードを新しく導入することで、乗り越え動作を表すアニメーションを状態遷移のグラフとして表現することができるようになりました。

アニメーション状態遷移の一つの状態として乗り越え動作を表すことにはいろいろなメリットがあります。

分岐を含む乗り越え動作も整理した形で表せるので、落下やキャンセルのような処理もシンプルに記述できます。また、アニメーションのコアシステムが持つイベントやブレンドといった機能を活用して、効率のよい実装が可能です。

また、乗り越え動作を専用の特殊なシステムによって実装せずに、通常の動作と同様のアニメーション状態の一つとすることで、同じツールを使って一貫したアニメーションデータとして作成・メンテナンスを行うことができます。

以上のようにして作成した乗り越え動作のアニメーションを、AIが制御する
キャラクタに行わせる際の手続きについて、次に説明します。




AIキャラクタに乗り越え動作を行わせるにあたっての方針は、乗り越えるかどうかはアニメーションシステムの側で全て判断して、AIシステムは細かいことを気にしなくてよいように実装する、というものです。

このような実装を行う目的は、AI システムは従来と同じようにキャラクタの経路計画を行って、そこに乗り越えられる障害物があるかどうかについて考える必要をなくすことにあります。動画では、AIが制御するキャラクターが画面手前にいるプレイヤーのところまで近寄ってくる様子を示しています。このとき、AI は途中にあるガードレールを飛び越えるような制御を行う必要はありません。

このような AI とアニメーションでの責任分担を実現するために、AI システムが参照するナビゲーションメッシュを作成するときに、乗り越えが可能な属性を持つ障害物の取扱いを工夫する必要があります。

動画で示した状況では、スライドに示すようにキャラクタが配置されています。

AIキャラクタの現在位置と移動目標地点の途中に、障害物となるガードレールがあります。ただし、このガードレールは乗り越えることが可能な属性を設定されています。

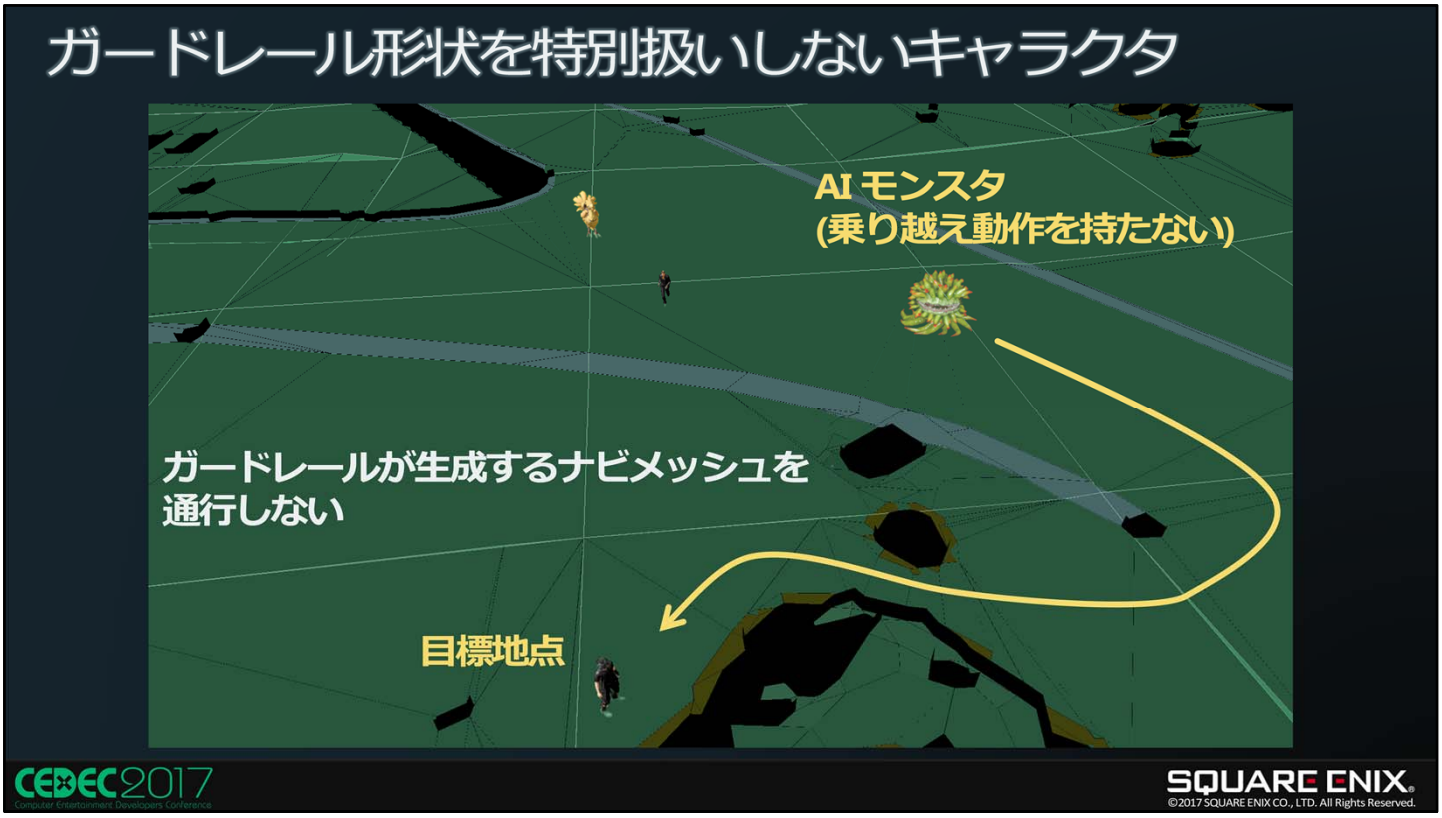
このような状況の地形からそのままナビゲーションメッシュを生成すると、ガードレールのところにスキマがあるナビメッシュとなります。

これは、ガードレールのコリジョンが平面的な移動では通行不能な形状になっているので、ナビメッシュとしては連続性がない形状として出力されるためです。

アニメーションシステムが乗り越え動作を実装している場合には、このガードレールは乗り越えることができます。

このようなアニメーションの性質を反映するために、ナビメッシュを生成するときに、ガードレールの形状を特別扱いして通過可能な地形であるとみなします。

このような特別扱いによって、ガードレールの周辺にもナビメッシュが生成され、AI キャラクタの現在位置と目標地点との間のナビメッシュが連続的につながった状態になるので、AI にとっては一直線に移動できる経路として認識されます。

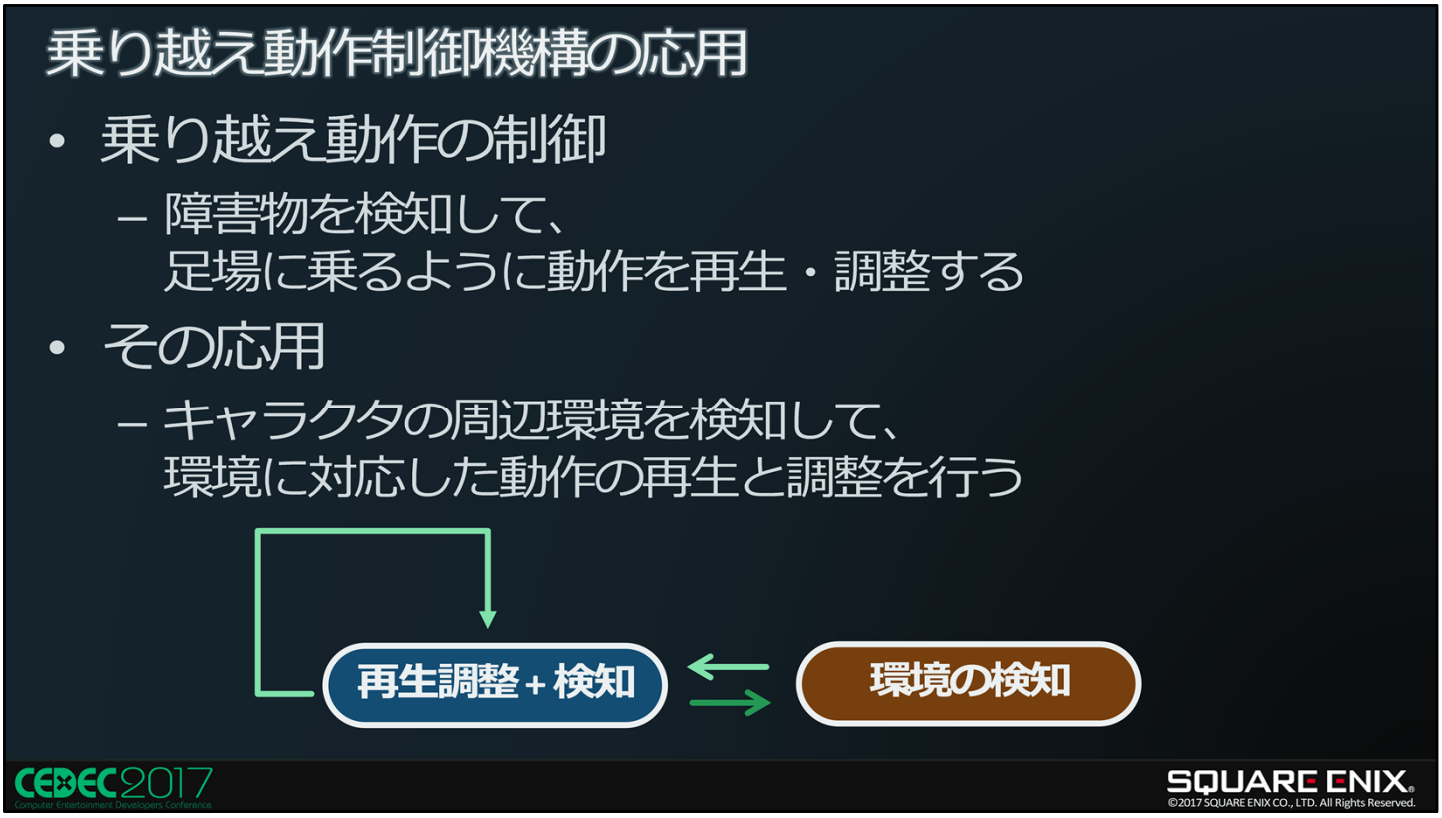



ただし、キャラクタによっては乗り越える動作を持たないモンスタのようなものもいます。

そういったキャラクタについては、移動経路を計画する時にガードレールが生成した部分のナビメッシュを無視することにより、ガードレールを乗り越えない移動経路を計画するようにしています。

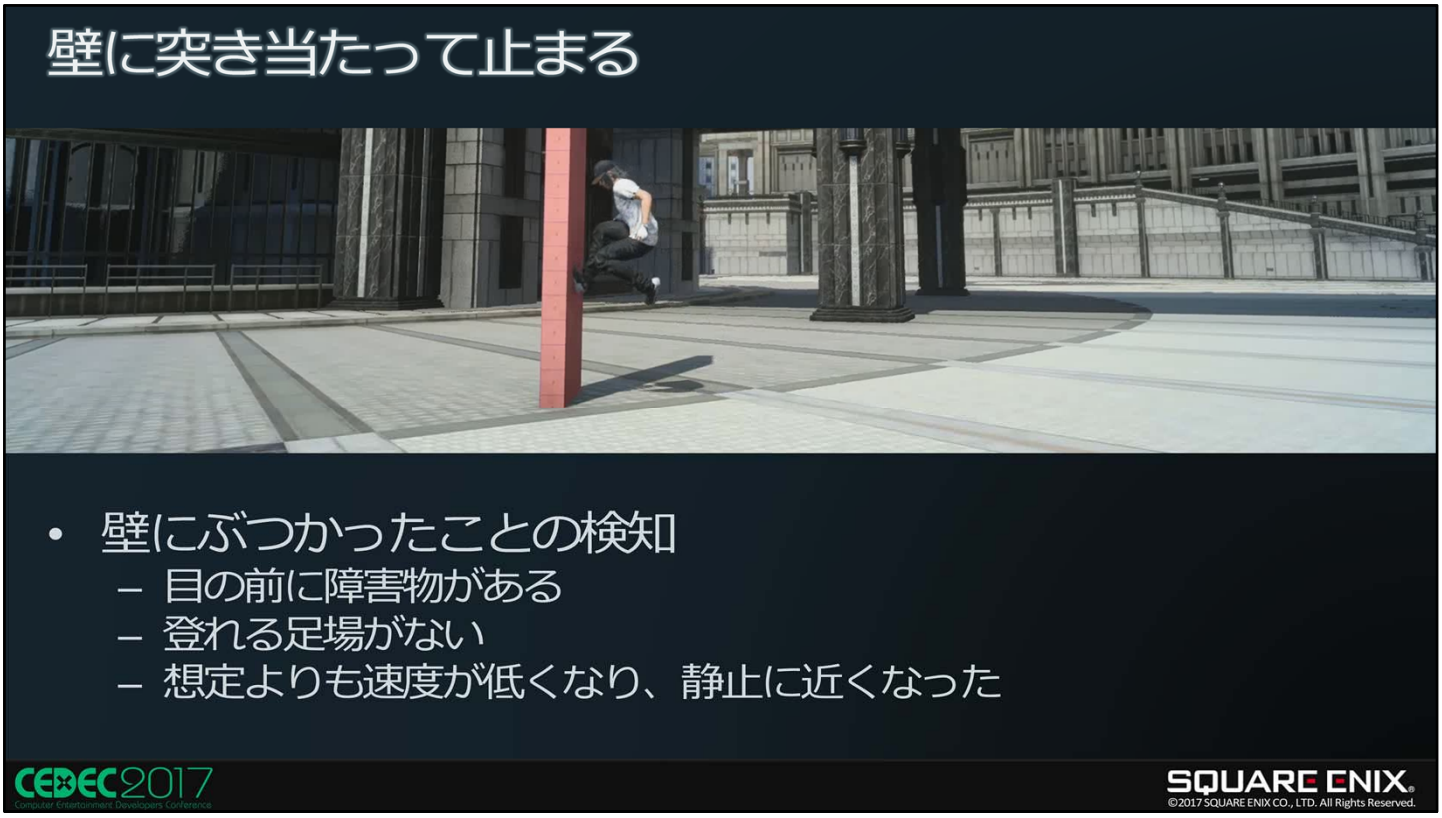
AI との連携については以上となります。

これまでに説明したシステムは障害物を乗り越える動作を制御するために開発したものですが、このシステムを乗り越え動作以外の動作の制御にも応用した事例について説明します。




乗り越え動作の基本的な動作は、障害物を検知して、その足場にちょうど乗るようにアニメーションクリップを再生し、調整するというものです。

これをより一般化して、キャラクタの周辺にある環境を検知して、検知された環境に対応した動作を再生する目的に応用することができます。




実際に応用した例としては、キャラクタが壁にぶつかって止まるときのリアクションがあります。

キャラクタの前に障害物があって登れないときや、動いているキャラクタが何かにぶつかって静止に近い状態になったことを検知して、ぶつかったリアクションの動作を再生する状態に遷移させます。

## 環境条件による壁止まり動作の抑制

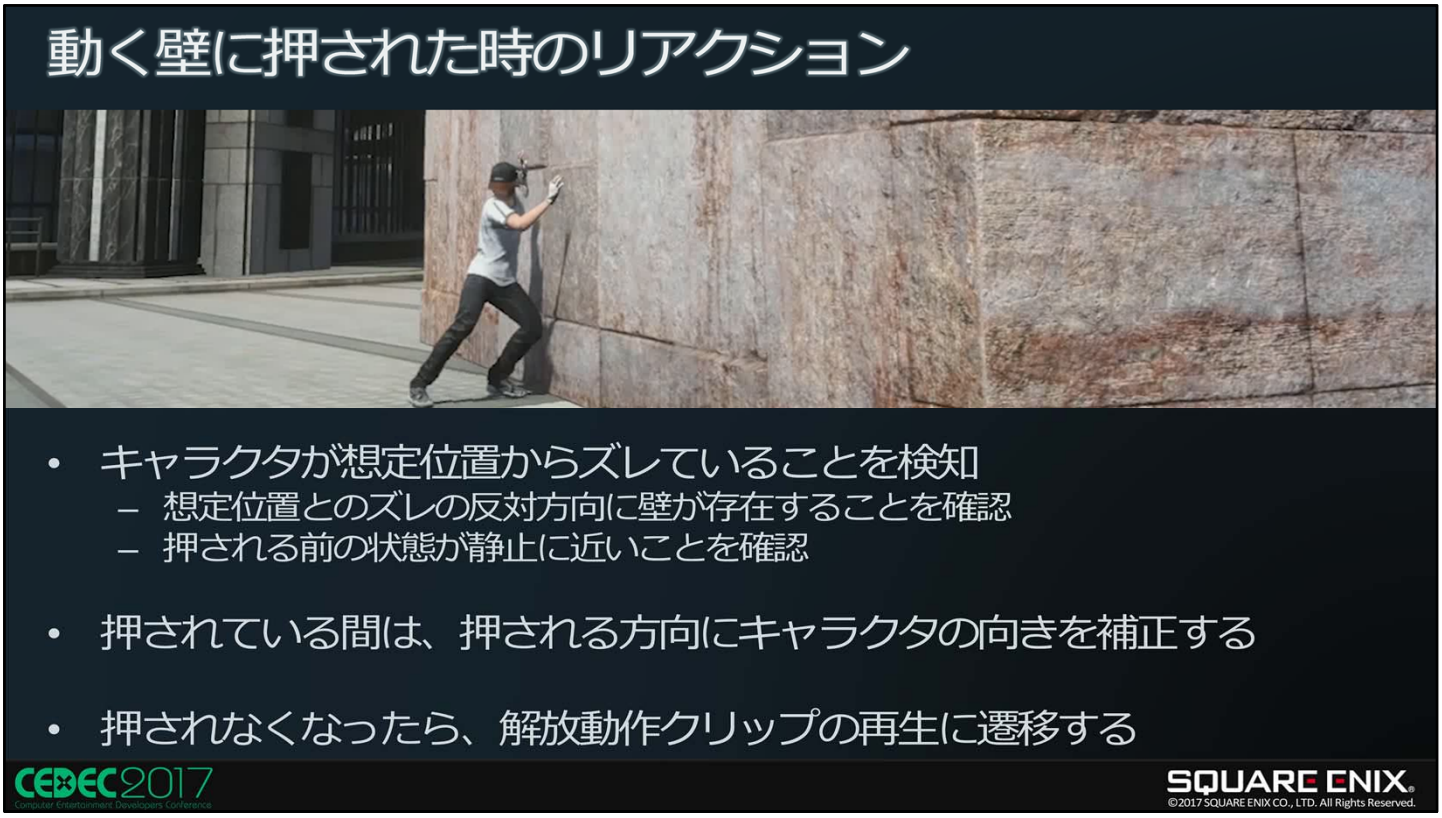
- 壁止まり動作アニメーションの再生が
  ゲームプレイの妨げにならないように配慮する
  - 操作レスポンスを優先するべき状況では
    壁止まり動作を抑制する
    - 戦闘中
  - 狭隘な空間内では、
    長尺の壁止まり動作を抑制する
    - ダンジョン内
    - 列車内

CEDEC2017
Computer Entertainment Developers Conference
SQUARE ENIX
©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

壁に当たって止まる動作に遷移するかどうかを判断するときには、ゲームプレイの妨げにならないようにキャラクタの置かれた状況をチェックしています。

キャラクタがバトル状態にあるときは、ユーザの操作に対する反応を最優先としたいので、壁で止まる動作の再生を行わないようにしています。

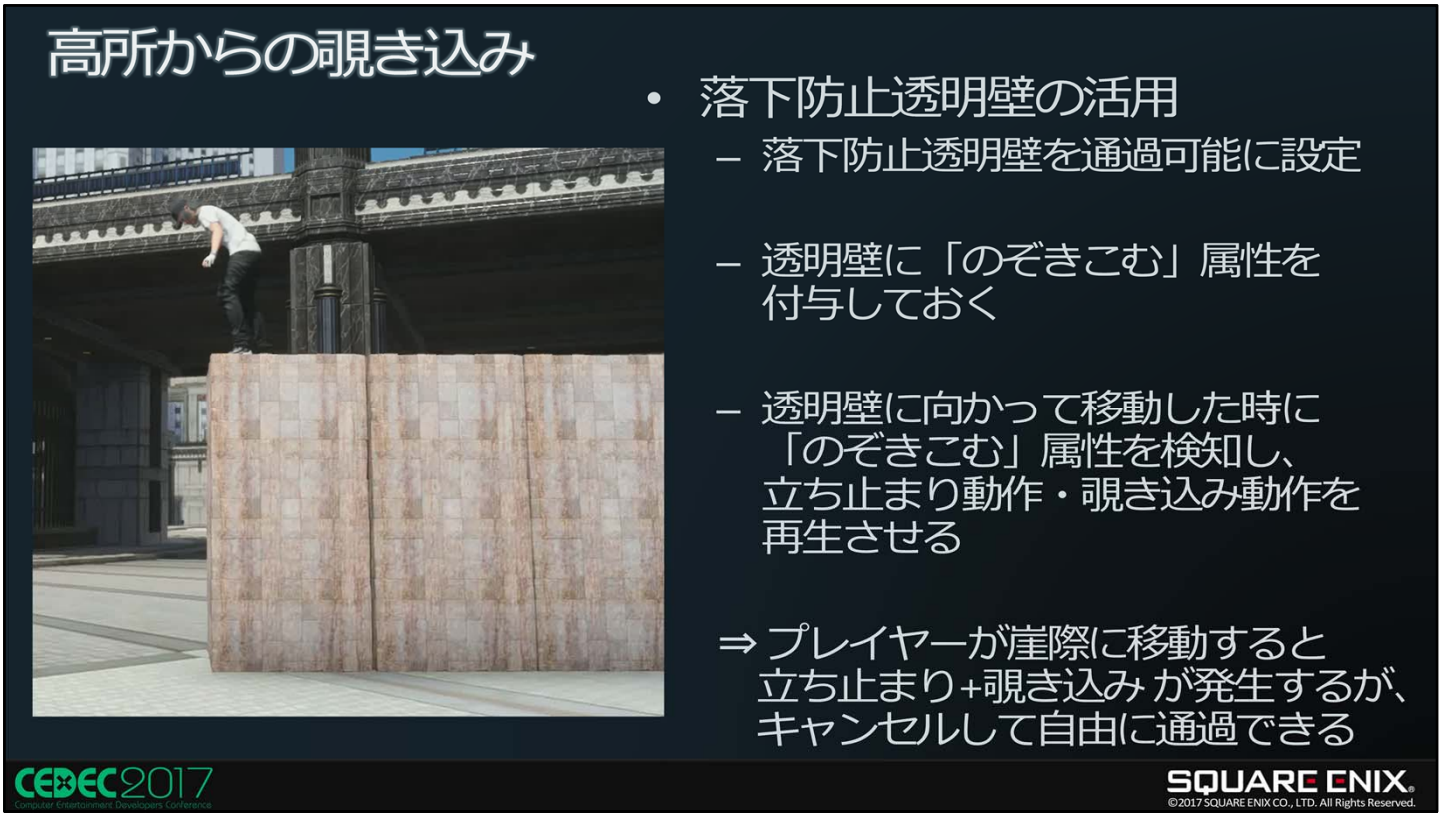
また、キャラクタが狭い空間にいるときは、壁を蹴って止まるような長い動作には遷移しないようになっています。このような状況としては、ダンジョンの通路や、列車の中にいる場合が該当します。




別の応用例としては、キャラクタが動く壁に押されたときのリアクション動作があります。

この場合には、キャラクタの位置が想定している位置からズレていることを検知してリアクションの動作に遷移します。また、遷移する際には押している壁が存在することを確認します。また、キャラクタが歩いている場合には歩いている状態を優先し、待機状態に近いときだけ押される動作に遷移するように調整しています。

アニメーションに対する制御としては、押されている状態が続いているあいだは、キャラクタの向きを押されている方向に補正します。また、押されなくなったらリアクションが終わったという動作の再生に遷移します。




また、別の応用として、高いところから下を覗く、といった特殊な動作の制御への応用も行いました。

この動作では、普段はキャラクタが崖から落下するのを防止している、見えないコリジョンの動作を変更しています。

通常この透明壁はキャラクタが通過できないように設定されていますが、これを通過できるように変更します。

その上で、透明壁の属性として、のぞきこむ動作の属性を与えておきます。

このような設定を行うことで、プレイヤーが透明壁に向かって移動したときに、のぞきこむ属性を検知して、キャラクタをガケの直前で立ち止まらせ、下を覗き込む動作の状態に遷移させることができます。

この透明壁そのものは自由に通過できるので、プレイヤーがガケに向かって移動する操作を行えば、覗き込む動作をキャンセルして落下することができるようになっています。

## 不自然な連続再生の防止

- 覗き込む透明壁のある崖際
  - 覗き込み再生終了後、すぐに再び透明壁が検知されると、覗き込み動作がいつまでも継続してしまう
- 連続再生防止のための検査項目
  - 前回再生した地点から一定距離の道のりを移動しているか?
  - 前回再生した時点から一定時間経過しているか?
- 検査を適用する動作種
  - 崖から覗き込む
  - 壁に突き当たって止まる
    (一度当たった場所では連続して立ち止まらない)

CEDEC2017 Computer Entertainment Developers Conference

SQUARE ENIX

©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

崖から下を覗き込む動作を透明壁の属性によって実装する場合、気をつける必要があるのは、覗き込み動作が終了したときに、すぐに透明壁の属性を検知すると、再びのぞきこむ属性が検知されて、覗き込む動作がいつまでも継続してしまう場合があるということです。

このような不自然な連続再生を防ぐために、覗き込む動作を行った場所と時間を記録しておき、前回覗き込んだところから一定距離離れるか、一定時間経過していなければ、覗き込み動作に遷移しないという工夫をしています。

また、壁に当たって止まる動作に関しても、同じ場所で連続してすぐに壁に当たる動作に遷移しないように設定を行っています。

全体のまとめにうつります。

## キャラクタアニメーションにおける乗り越え動作

- 移動の制御としての乗り越え動作は平易な処理
  - 移動距離と速度の調整
  - 移動方向の調整
- 関連システムへの接続には工夫が必要
  - 足場データの持ち方・足場位置の検出
  - AI による経路計画との連携
  - アニメーション状態遷移への統合

CEDEC2017 Computer Entertainment Developers Conference

SQUARE ENIX

©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

本発表では、キャラクタが障害物を乗り越える動作を制御するために実装した方法について説明しました。

障害物を乗り越えるようにキャラクタの位置や速度を調整する処理はシンプルなものですが、足場を検出するための方法や、アニメーションの状態遷移として実装するときに、データの作成やゲームの実行を効率よく行う工夫が必要だったということについて説明しました。

## 状態遷移とブレンドによる乗り越え動作の実装

- アニメーションの状態としての乗り越え動作
  - 既存の移動状態・待機状態の拡張として表現できる
  - 状態遷移はシンプルに整理できる
  - より一般化して、壁止まり等の動作も取り扱える
- 専用ブレンドノードによる実装
  1. 乗り越えの足場を見つけるブレンドノード
  2. 足場に対する移動量を調整するブレンドノード

CEDEC2017 Computer Entertainment Developers Conference

SQUARE ENIX

©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

乗り越え動作を制御するシステムは、状態遷移とブレンドツリーに基づくアニメーションシステムの一部として実装しました。

この実装では、２つのブレンドノードを乗り越え動作の制御のために実装しました。これにより、既存のアニメーショングラフでの移動状態や待機状態に状態を付け加える形で乗り越え動作を表現し、シンプルに整理することが可能になりました。

また、より一般化して、壁に突き当たって止まる動作などにも応用した例について説明しました。

## 今後の展望 (1/3)

- 動作種の拡充
  - より複雑な乗り越え
    - (例) 横にある壁面を蹴って登る
  - 断崖からの飛び降り
    - (例) ちょうど踏み切る場所を検知する
  - その他
    - (例) 飛び石渡り

CEDEC2017 Computer Entertainment Developers Conference
SQUARE ENIX ©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

今回実装した乗り越え動作の制御システムについて、今後の展望についてお話します。

まず、取り扱える動作の種類について、より複雑な乗り越えや、乗り越え以外の動作への応用を広げたいと考えています。

その例としては、たとえば障害物を乗り越えるときに、まっすぐ登るだけではなく、横に壁があればそちらを利用して登るような動作が考えられます。

## 今後の展望 (2/3)

- 静止していない物体の乗り越え
  - 移動調整ノードの拡張が必要
- 詳細な足場形状データの利用
  - 乗り越え動作専用のナビゲーション用データ
  - 窓枠や木の枝など、物体の特定部分だけを足場として利用したいときに必要

CEDEC2017 Computer Entertainment Developers Conference

SQUARE ENIX

©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

また、現在の実装では、乗り越える動作の対象となる障害物は移動しておらず、静止しているものという仮定がありますが、動いている障害物もうまく乗り越えられるような拡張が必要だと考えています。

FFXV では障害物について、物体の単位で乗り越えられるかどうかを設定していましたが、物体の特定の部分だけを足場とすることを目的として、詳細な足場形状のデータも同時に参照できるような仕組みも必要だと考えています。

## 今後の展望 (3/3)

- 乗り越え状態に遷移する前後での足の位置合わせ
  - 遷移直前での足の構え状態を考慮
  - アニメーションコアの左右反転機能の活用
- 乗り越えや壁止まり時の姿勢を微調整
  - 検知した障害物位置に足裏や手をよりフィットさせる
  - Inverse Kinematics 処理の適用

CEDEC2017 Computer Entertainment Developers Conference
SQUARE ENIX ©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

現在のアニメーショングラフでは、ダッシュ状態から乗り越え状態に遷移するとき、足の構えが正しくつながっているかどうかは特に考慮せずに、障害物との距離だけで遷移のタイミングを決定しています。

ゲームプレイにスローかかったり、ゆっくり歩いている状態から乗り越え動作に遷移したりすると足の位置が不連続になる不自然さが目立ってくるので、アニメーションコアのポーズ左右反転機能を活用するなどして、足の位置が遷移の前後で不連続になるのを緩和する仕組みが必要だと考えています。

また、これとは別に、壁に当たって止まるような動作を行うとき、検知した壁にたいして、添えた手や足がよりフィットするように IK 処理を用いて補正を行うと、より品質の高い動作を生成できると考えています。

## 乗り越え動作アニメーションの制御に関係する発表

- “AI Postmortems: Assassin's Creed III, XCOM: Enemy Unknown, and Warframe”, GDC 2013
- “In Your Hands: The Character (of Watch Dogs)”, GDC 2015
- “Player Traversal Mechanics in the Vast World of 'Horizon: Zero Dawn'”, GDC 2017

CEDEC2017 Computer Entertainment Developers Conference

SQUARE ENIX
©2017 SQUARE ENIX CO., LTD. All Rights Reserved.

乗り越え動作の制御に関する発表は以上となります。

この発表では、ご来場のみなさまが、開発中のゲームのアニメーションシステムで障害物を乗り越える動作をサポートしたくなったときに、データやプログラムにどのような拡張を行えばよいか、という見通しをつけることができるようになることを目標にしていました。

FFXV で行った工夫についてはこの発表でできるだけ説明しましたが、乗り越え動作の制御というテーマについてはこれまでにもいくつか発表が行われているので、参考にしていただけると思います。

FM&A
IUZ
Q & A
• ASSASSIN'S CREED は UBISOFT ENTERTAINMENT の商標または登録商標です。
• XCOM はテイクツー インタラクティブ ソフトウェア インクの商標または登録商標です。
• Warframe はデジタル エクストリームズ リミテッドの商標または登録商標です。
• WATCH DOGS は UBISOFT ENTERTAINMENT の商標または登録商標です。
• Horizon Zero Dawn は株式会社ソニー・インタラクティブエンタテインメントの商標または登録商標です。
• その他掲載されている会社名、商品名は、各社の商標または登録商標です。
CEDEC2017
Computer Entertainment Developers Conference
SQUARE ENIX
©2017 SQUARE ENIX CO., LTD. All Rights Reserved.