マインクラフト・プログラミング応用

第4回　イベント処理

☆イベント処理とは
☆花火を上げてみよう
☆爆発するニワトリ
☆ScriptCraft のAPI一覧
☆イベント処理サンプル集

2017.08.09

鎌倉シチズンネット（KCN）

イベント処理 (1)

- イベント処理とは

  イベント処理とは、あるイベント（事象）が発生したときに行う処理のことで、「あるイベント（事象）が発生したときにどういう処理（関数）を実行するか」を記述します。

  ブロックが破壊されたというイベントが発生したら、「あなたがブロックを破壊した」というメッセージを表示するときは次のように記述します。

```
function myBlockBreakHook( event ){
    var breaker = event.player;
    echo( breaker, 'You broke a block');
 }
events.blockBreak( myBlockBreakHook );
```

  上記のmyBlockBreakHook　は適当な名前の関数でよく、イベントのコールバック関数あるいはフック関数と呼ばれます。

イベント処理 (2)

- イベント処理の取り消し

  イベント処理を一度だけ実行して、その後は実行しないようにするには、「unregister()」というメソッドを使用します。

```
function myBlockBreakHook( event ){
    var breaker = event.player;
    echo( breaker, 'You broke a block');
    this.unregister();
 }
events.blockBreak( myBlockBreakHook );
```

  コールバック関数の外部で取り消すときは次のように記述します。

```
var  myBlockBreakListener = events.blockBreak( myBlockBreakHook );
・・・
myBlockBreakListener.unregister();
```

イベント処理 (3)

- イベント処理の練習問題
  プレイヤーがゲームに参加してきたら、「Hello, { プレイヤー名 }」と表示するイベント処理を記述してみよう！

  events.playerJoin ( ); を使います。
  echo (player, 'Hello, ' + player.name); でメッセージを表示します。

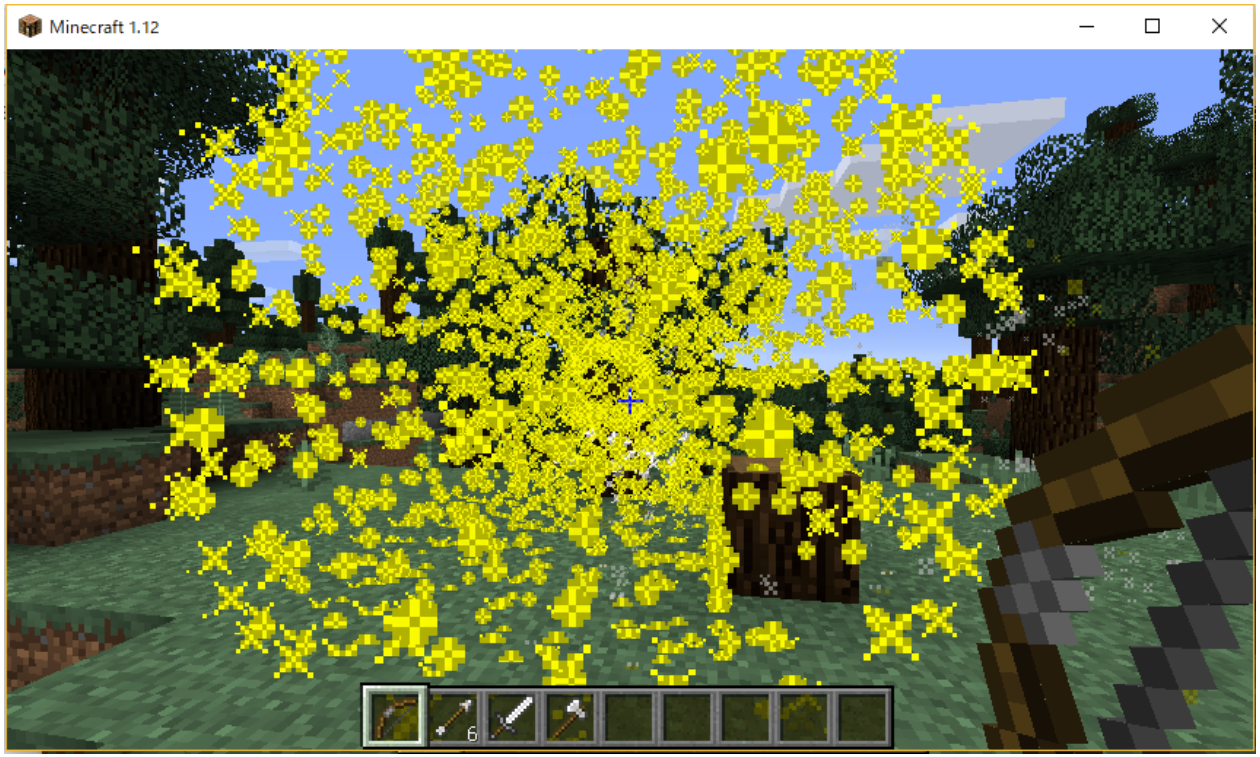
イベント処理 (4)

- 各プレイヤーが壊したブロックの数をカウントしよう

```
var breaks = {}; /* every time a player joins the game reset their block-break-count to 0 */
function initializeBreakCount( event ){
    breaks[event.player.name] = 0;
}
events.playerJoin( initializeBreakCount ); /* every time a player breaks a block increase their block-break-count */
function incrementBreakCount( event ){
    breaks[event.player.name] += 1; // add 1
    var  breakCount = breaks[event.player.name];
    echo( event.player, 'You broke ' + breakCount + ' blocks');
}
events.blockBreak( incrementBreakCount );
```

花火を上げてみよう (1)

- 矢が当たったところに花火を上げてみよう
  events.projectileHit ( ) というイベントを使用します。



（クリックすると動画になります）

花火を上げてみよう(2)

- myFireworks.js のコードです

```
var fireworks = require('fireworks'),
    bkArrow = org.bukkit.entity.Arrow,
    bkPlayer = org.bukkit.entity.Player;

exports.myFirework = function(){

 console.log("myFirework: [Entering  Main]");

 function onBukkitArrowHit( event ) {
  var projectile = event.entity,
      shooter = projectile.shooter,
      fireworkCount = 5;

  function launch(){
   fireworks.firework( projectile.location );
   if ( --fireworkCount ) {
    setTimeout( launch, 2000 );
   }
  }
  if (projectile instanceof bkArrow
      && shooter instanceof bkPlayer) {

   projectile.remove();
   launch();
  }
  this.unregister();
 }
 events.projectileHit( onBukkitArrowHit );
}
```

myFireworks.js をc:¥Users¥（名前）¥spigot¥scriptcraft¥plugins¥{Minecraft のユーザ名} に保存し、チャット欄に次のように入力します。

/js refresh()

/js myFirework()

弓矢で矢をどこかにあてると、花火が上がります。

爆発するニワトリ (1)

- 爆発するニワトリ
  ニワトリに矢が当たると2秒後に大爆発が発生します。

爆発するニワトリ(2)

- 爆発するニワトリのコードです

下記のコードをc:¥Users¥（名前）¥spigot¥scriptcraft¥plugins¥{Minecraft のユーザ名} に保存し、チャット欄に次のように入力します。
/js refresh()
/js exploding_chickens()

```
exports.exploding_chickens = function(){
var bkPlayer = org.bukkit.entity.Player;
var bkChicken = org.bukkit.entity.Chicken;
// ニワトリのスポーン
self.location.world.spawnEntity(self.location,org.bukkit.entity.EntityType.CHICKEN)
// The function which will run when we load this module
var _loadMod = function()
{
  // Announce ourselves to the console
  console.log("Exploding Chickens: [Loading ScriptCraft Mod]");  //  コンソールへ表示

  // Tie our code into the event that fires every time one entity damages another
  events.entityDamageByEntity(_entityDamageByEntity);  //  イベント処理の登録
}
```

爆発するニワトリ(3)

- exploding_chickens.js のコードです（続き）

```
// The code that we want to run each time one entity damages another
var _entityDamageByEntity = function(event)   //  イベント発生時に実行される処理
{
  // Find out, from the event, who's getting damaged and who did the damage
  var damagedEntity = event.getEntity();
  var damagingEntity = event.getDamager();

  // If it's a chicken getting damaged by a player, game on...
  if (damagedEntity instanceof bkChicken && damagingEntity instanceof bkPlayer)
  {  //  プレイヤーがニワトリにダメージを与えたら
    // Announce in the console that we've detected a player damaging a chicken
    console.log("Exploding Chickens: [A player damaged a chicken]");  //  コンソールに表示
```

爆発するニワトリ(4)

- exploding_chickens.js のコードです（続き）

```
// Schedule a task to run in five seconds.  //  5秒後に実行
    server.scheduler.scheduleSyncDelayedTask(__plugin, function()
    {
      // Check to see if the damage brings the chicken's health
      // down to, or below, zero. If so, it's dead...
      if (damagedEntity.getHealth() - event.getDamage() <= 0)  //  ニワトリの体力が無くなったら
      {
        // Get the chicken's location
        var loc = damagedEntity.location;

        // Create an explosion at the chicken's location.
        // A big one...
        loc.world.createExplosion(loc, 10.0);   //  ニワトリがいた所で大爆発
      }
    }, 20 * 5);
  }
}
// Run this script as soon as the file's loaded
_loadMod();
}
```

爆発するニワトリ(5)

- 練習問題

  ニワトリが死んだときに、ニワトリを爆発させる代わりに、ニワトリが居たところに、雷を落としてみよう。雷を落とすときには次のように記述します。

  ```
  entity.location.world.strikeLightning(entity.location);
  ```

ScriptCraft のAPI一覧(0)

- メソッド

  天気が雨のときに次のコマンドを入力すると、雷が鳴り稲妻が光るようになる。

  ```
  /js  self.world.setThundering ( true )
  ```

  このように記述すると、Spigot API Reference にあるSpigot のメソッドを呼び出すことができる。https://hub.spigotmc.org/javadocs/spigot/org/bukkit/World.html

ScriptCraft のAPI一覧(1)

- 主なオブジェクトのプロパティ（フィールド）

オブジェクト	プロパティ	意味
event	player	イベントを発生させたプレイヤー
event	block	イベントが発生したブロック
event	entity	イベントが発生したエンティティ
world	thundering	雷雨かどうか
world	storm	嵐かどうか
player	allowFlight	飛行を許可するか
player	onGround	地上にいるか（空中ではない）
player	sneaking	スニークしているか
player	name	プレイヤー名

ScriptCraft のAPI一覧(2)

- 主なオブジェクトのメソッド(1)

オブジェクト	メソッド	意味
event	getEntity()	イベントが発生したエンティティを得る
event	getDamager()	ダメージを与えたエンティティを得る
event	getDamage()	ダメージの大きさを得る
block	getLocation()	ブロックの位置を得る
block	getType()	ブロックのタイプを得る
block	getX(), getY(),getZ()	ブロックの座標を得る
entity	getType()	エンティティのタイプを得る
entity	getLocation()	エンティティの位置を得る
location	getBlock()	ロケーションにあるブロックを得る
location	getWorld()	ロケーションのワールドを得る

ScriptCraft のAPI一覧(3)

- 主なオブジェクトのメソッド(2)

オブジェクト	メソッド	意味
world	createExplosion(location, power)	爆発を発生させる
world	setStorm(true)	嵐を発生させる
world	hasStorm()	嵐かどうか
world	setThundering(true)	雷を発生させる
world	isThundering()	雷かどうか
world	dropItem(location, item)	指定した場所にアイテムを落とす
server	addRecipe(recipe)	レシピを追加する

イベント処理サンプル集(1)

処理内容をクリックすると、JavaScript のソースをダウンロードすることができます。

イベント	処理内容
playerInteract	プレイヤーがブロックを右クリックしたとき、ダイヤモンドのブロックに変更する
playerInteract	プレイヤーがブロックを右クリックしたときブロックを消す
playerInteract	プレイヤーがブロックを右クリックしたとき隣合っているブロックをすべて消す
playerJoin	プレイヤーがログインするときウェルカムメッセージを表示する
playerQuit	プレイヤーがログアウトするとき
playerDeath	プレイヤーが死ぬとき、プレイヤーをリスポーンさせる
playerRespawn	プレイヤーがリスポーンするとき
playerInteractEntity	プレイヤーがエンティティを右クリックすると、エンティティが消える(remove)

イベント処理サンプル集(2)

処理内容をクリックすると、JavaScript のソースをダウンロードすることができます。

イベント	処理内容
blockPlace	ブロックがプレイヤーによって置かれたとき、置かれたブロックをダイヤモンドのブロックに変更する
blockPlace	ブロックがプレイヤーによって置かれたとき、置かれたブロックを削除する（空気にする）
blockPlace	ブロックがOP権限のないプレイヤーによって置かれたとき、ブロックを置かないようにする
blockPlace	TNTブロックがOP権限のないプレイヤーによって置かれたとき、TNTブロックを置かないようにする
blockBreak	ブロックがプレイヤーによって壊されたとき、壊されたブロックを復元する
blockBreak	ブロックがプレイヤーによって壊されたとき、アイテムのクッキーを2枚落とす
blockBreak	ブロックがOP権限のないプレイヤーに壊されないように

イベント処理サンプル集(2)

処理内容をクリックすると、JavaScript のソースをダウンロードすることができます。

イベント	処理内容
entitySpawn	エンティティがスポーンしたとき、同じエンティティをもう1つスポーンさせる（双子にする）
entityDeath	エンティティが死ぬとき

イベント処理サンプル集(3)

処理内容をクリックすると、JavaScript のソースをダウンロードすることができます。

イベント	処理内容
projectileHit	飛び道具（矢など）が当たったとき、花火を上げる
projectileHit	飛び道具（矢など）が当たったとき、雷を落とす（木の葉が燃えます）
projectileHit	飛び道具（矢など）が当たったとき、雷雨にする（不安定）
projectileHit	飛び道具（矢など）が当たったとき、爆発させる
projectileHit	飛び道具（矢など）が当たったとき、プレイヤーを当たった位置までテレポートさせる
projectileHit	飛び道具（矢など）が当たったとき、木のブロックを置く
entityDamageByEntity	矢がニワトリに当たると大爆発が発生する