singletonのサンプル？

シングルトンパターン比較

シングルトンパターンじゃなく作ったPlayerクラスをマウスクリックするたびにnewする動画
staticなメンバ変数`playerCount`がクリックするたびに増えて、自分の番号＝playercountになるよ。

つまり、クリックするたびに、theMain.cppで宣言されている`vector<Hero*> heroes`に新しく生成されたPlayerのオブジェクトが追加されている。

https://www.youtube.com/embed/Q1kJWBAbn7E?si=gY9Rfz0x15K6wo7V

適当なソースコード

```
//Main.cpp
//省略
namespace
{
 const int BGCOLOR[3] = {0, 0, 0}; // 背景色{ 255, 250, 205 }; // 背景色
 int crrTime;
 int prevTime;
 vector<Hero*> heroes; // プレイヤーのポインタを格納するベクター
}

//省略
//ここにやりたい処理を書く
if ((oldMouseInput & MOUSE_INPUT_LEFT) == 0 && (mouseInput & MOUSE_INPUT_LEFT) != 0)
{
 int x, y;
 GetMousePoint(&x, &y);
 Hero *p=new Hero();
 p->SetPosition(x, y);
 p->SetPosition(x, y);
 heroes.push_back(p);
}


for (auto& hero : heroes)
{
 hero->Update();
 hero->Draw();
}
ScreenFlip();
//省略
```

```
#pragma once
class Hero
{
private:
 int posx_, posy_; // プレイヤーの座標
 int myNumber_;// プレイヤーの番号
 int hImage_; // プレイヤーの画像
 static int playerCount_; // プレイヤーの数
public:
 Hero();
 ~Hero();
 void SetPosition(int x, int y);
 void Update();
```

```
 void Draw();
};

Hero::Hero()
 : posx_(0), posy_(0), myNumber_(0), hImage_(-1)
{// プレイヤーの画像を読み込む
 hImage_ = LoadGraph("Assets/tiny_ship5.png");
 if (hImage_ == -1)
 {
  // 画像の読み込みに失敗した場合の処理
 }
 playerCount_++;
 // プレイヤーの番号を設定
 myNumber_ = playerCount_;
}

Hero::~Hero()
{
}

void Hero::SetPosition(int x, int y)
{
 posx_ = x;
 posy_ = y;
}

void Hero::Update()
{
}

void Hero::Draw()
{
 DrawGraph(posx_, posy_, hImage_, TRUE);
 DrawFormatString(posx_+32+2, posy_-2, GetColor(255, 255, 255), "Player %d", myNumber_);
}
```

singletonパターンで書いてみる

singletonにすると、うまくプログラムが動いていれば、実行プログラム内でPlayerのインスタンスは一度newされたらその一つだけになる。（はず）
上の通常の実装と同じように、クリックするたびに、インスタンスのポインタをリストに入れていくとインスタンスがプログラム内で一つしかないなら、同一のアドレスが何度もリストに登録されているはずである。

またまた適当なソースコード（singleton風）

```
//theMain.cpp
//省略
namespace
{
 const int BGCOLOR[3] = {0, 0, 0}; // 背景色{ 255, 250, 205 }; // 背景色
 int crrTime;
 int prevTime;
 vector<Hero*> heroes; // プレイヤーのポインタを格納するベクター
}
//省略
 while (true)
 {
  oldMouseInput = mouseInput;
  mouseInput = GetMouseInput();

//省略
  //ここにやりたい処理を書く
  if ((oldMouseInput & MOUSE_INPUT_LEFT) == 0 && (mouseInput & MOUSE_INPUT_LEFT) != 0)
  {
   int x, y;
   GetMousePoint(&x, &y);
   Hero* p = &(Hero::GetInstance());
   p->SetPosition(x, y);
```

```
   heroes.push_back(p); // ベクターにポインタを追加（シングルトンなら同じアドレスしか入っていかないはず）
  }
//省略
  for (auto& hero : heroes)
  {
   hero->Update();
   hero->Draw();
  }
  ScreenFlip();

 }
  //省略
```

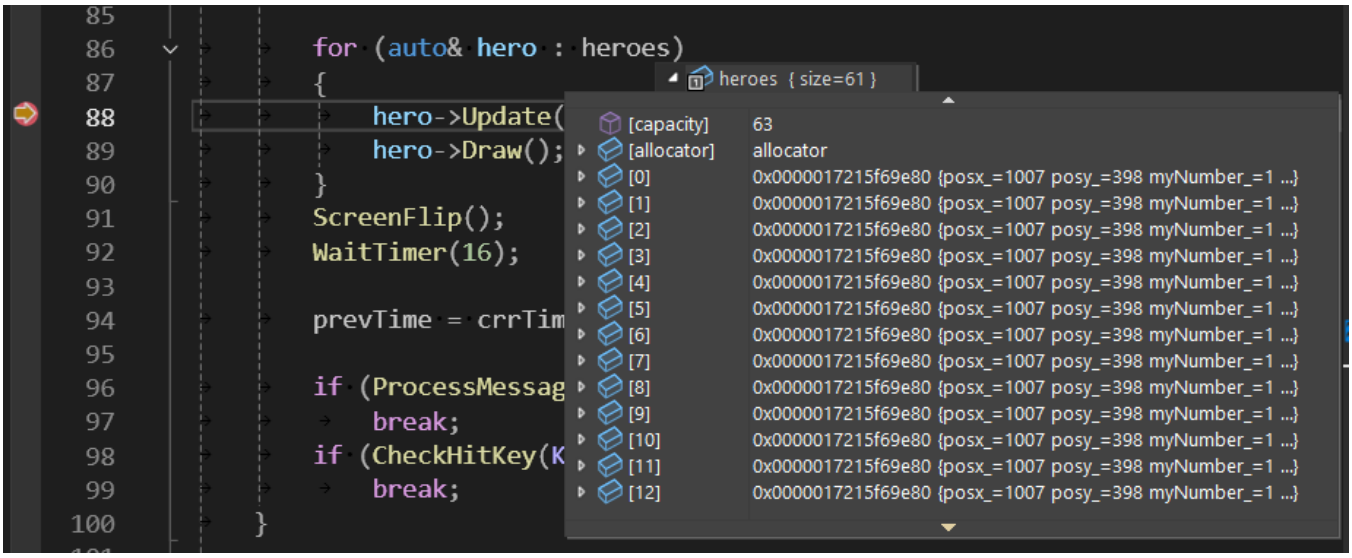
```
#pragma once
class Hero
{
private:
 int posx_, posy_; // プレイヤーの座標
 int myNumber_;// プレイヤーの番号　ずっと0のまま（シングルトンだから）
 int hImage_; // プレイヤーの画像
 static inline int playerCount_ = 0; // プレイヤーの数（シングルトンならふえないはず）
 static inline Hero* instance_ = nullptr; // 唯一のインスタンス
 Hero(); // コンストラクタをprivateにする
 ~Hero();
public:
 // シングルトンパターンを使用して、唯一のインスタンスを取得する
 static Hero& GetInstance();
 //インスタンスを削除する
 static void DeleteInstance();
 void SetPosition(int x, int y);
 void Update();
 void Draw();
 //コピーと代入を禁止する
 Hero(const Hero&) = delete;
 Hero& operator=(const Hero&) = delete;
};
Hero::Hero()
 : posx_(0), posy_(0), myNumber_(0), hImage_(-1)
{// プレイヤーの画像を読み込む
 hImage_ = LoadGraph("Assets/tiny_ship5.png");
 if (hImage_ == -1)
 {
  // 画像の読み込みに失敗した場合の処理
  //おサボりでかかない（みんなは書いてね）
 }
 playerCount_++;
 myNumber_ = playerCount_;  // インスタンス番号として記録
}

Hero::~Hero()
{
 // 画像の解放
 if (hImage_ != -1)
 {
  DeleteGraph(hImage_);
  hImage_ = -1;
 }
}

Hero& Hero::GetInstance()
{
 if (instance_ == nullptr)
 {
  instance_ = new Hero();
 }
 return *instance_;
}
```

```
void Hero::DeleteInstance()
{
}

void Hero::SetPosition(int x, int y)
{
 posx_ = x;
 posy_ = y;
}

void Hero::Update()
{
}

void Hero::Draw()
{
 DrawGraph(posx_, posy_, hImage_, TRUE);
 DrawFormatString(posx_+32+2, posy_-2, GetColor(255, 255, 255), "Player %d", myNumber_);
}
```

結果

https://www.youtube.com/embed/OCuqXxRyNhk?si=Pk0AE4Uf6pa8CR0e

一応、念のため、何度かクリックしたあとのリストの中身を表示してみると。。。



全部同じアドレスが入ってる。無駄な描画してるね笑
でも、クリックしたときにGetInstanceを呼んでも一度インスタンスが生成されてしまえば、同じインスタンスのアドレスを返してくれることがわかった（singletonとして動いているっぽい）
めでたしめでたし。

Revision #3
★Created 18 May 2025 15:04:16 by youe2
Updated 2 June 2026 19:54:18 by youe2