ReStudyC++_12

当てた Enemy を Player に変更する

```
/*
C++基本の総復習オマケ4（コンポーネント指向プログラミングの基礎）
*/

//【今回のコード　ー　仕様変更の例】
//当てたEnemyをPlayerに変更する

#include "Component.h"
#include <list>
#include <Windows.h>

ScreenBuffer g_ScreenBuffer;
char InputData::Buffer = 0;

class Object;
class Player;

class Component
{
protected:
public:
	Component() {}
	virtual ~Component() {}
	Object* Parent;
	virtual void Start() {}
	virtual void Update() {}
	virtual void Draw() {}
};

class Object
{
public:
	Object() {}
	~Object() {
		for (auto com : ComponentList)
			delete com;
	}
```

```
	std::list<Component*> ComponentList;
	void Update()
	{
		auto buff = ComponentList;
		for (auto com : buff)
			com->Update();
	}
	void Draw()
	{
		for (auto com : ComponentList)
			com->Draw();
	}

	//オブジェクトが持っているコンポーネントを取得
	template<class T>
	T* GetComponent()
	{
		for (auto com : ComponentList) {
			T* buff = dynamic_cast<T*>(com);
			if (buff != nullptr)
				return buff;
		}
		return nullptr;
	}

	//オブジェクトが持っているコンポーネントを追加
	template<class T>
	T* AddComponent()
	{
		T* buff = new T();
		buff->Parent = this;
		ComponentList.push_back(buff);
		buff->Start();
		return buff;
	}
};

//オブジェクトのリストを定義
std::list<Object*> g_ObjectList;

//場所を示すコンポーネント
```

```
class Position : public Component
{
public:
	int x, y;
};

//敵コンポーネント
class Enemy : public Component
{
	Position* pos = nullptr;
public:
	void Update()
	{

	}
	void Draw()
	{
		if (pos == nullptr)
			pos = Parent->GetComponent<Position>();
		g_ScreenBuffer.buffer2[pos->x][pos->y] = 'E';
	}
};

//【ここだけを変えればいい】
//弾コンポーネント
class Bullet : public Component
{
	Position* pos = nullptr;
public:
	void Update()
	{
		if (pos == nullptr)
			pos = Parent->GetComponent<Position>();
		pos->x--;
		//画面外に消えてたら自分を消す
		if (pos->x < 0)
		{
			Parent->ComponentList.remove(this);
			delete this;
			return;
		}
```

```
                //ObjectListからEnemyを検索して当たり判定を行い削除もする。
                auto buff = g_ObjectList;
                for (auto obj : buff)
                {
                        //全体からEnemyを探す
                        Enemy* enemy = obj->GetComponent<Enemy>();
                        if (enemy == nullptr)
                                continue;
                        if (obj->GetComponent<Position>()->x == pos->x && obj->GetComponent<Position>()->y == pos->y) {
                                //オブジェクトを削除しない
                                //g_ObjectList.remove(obj);
                                //delete obj;

                                //敵コンポーネントを削除
                                obj->ComponentList.remove(enemy);
                                delete enemy;

                                //プレイヤーコンポーネント追加
                                obj->AddComponent<Player>();
                        }
                }
        }
        void Draw()
        {
                if (pos == nullptr)
                        pos = Parent->GetComponent<Position>();
                g_ScreenBuffer.buffer2[pos->x][pos->y] = 'b';
        }
};

class Player : public Component
{
        Position* pos = nullptr;
public:
        void Start()
        {
                if (pos == nullptr)
                        pos = Parent->GetComponent<Position>();
        }
        void Draw()
        {
```

```
                if (pos == nullptr)
                        pos = Parent->GetComponent<Position>();
                g_ScreenBuffer.buffer2[pos->x][pos->y] = 'p';
        }

        void Update()
        {
                if (pos == nullptr)
                        pos = Parent->GetComponent<Position>();
                //移動
                if (InputData::KeyCheck('d') && pos->y < SCREE_NLENGTH - 1)
                        pos->y++;
                if (InputData::KeyCheck('a') && pos->y > 0)
                        pos->y--;
                if (InputData::KeyCheck('s') && pos->x < SCREE_NLENGTH - 1)
                        pos->x++;
                if (InputData::KeyCheck('w') && pos->x > 0)
                        pos->x--;

                //球発射
                if (InputData::KeyCheck(' '))
                {
                        Object* obj = new Object;
                        Position* posb = obj->AddComponent<Position>();
                        obj->AddComponent<Bullet>();
                        posb->y = pos->y;
                        posb->x = pos->x - 1;
                        g_ObjectList.push_back(obj);
                }
        }
};

int main()
{
        //追加
        Object* obj = new Object;
        Position* pos = obj->AddComponent<Position>();
        pos->x = 5; pos->y = 8;
        obj->AddComponent<Player>();
        g_ObjectList.push_back(obj);

        for (int i = 0; i < 10; i++)
```

```
	{
		obj = new Object;
		pos = obj->AddComponent<Position>();
		pos->x = 1;
		pos->y = i;
		obj->AddComponent<Enemy>();
		g_ObjectList.push_back(obj);
	}

	while (!InputData::KeyCheck('p'))
	{
		//画面の初期化
		system("cls");
		g_ScreenBuffer.Clear();
		InputData::Update();

		//実際の処理
		//Update中にObjectListがいじられてイテレーションバグるのを回避
		auto buff = g_ObjectList;
		for (auto obj : buff)
			obj->Update();

		for (auto obj : g_ObjectList)
			obj->Draw();

		//Buffer表示
		printf("%s", g_ScreenBuffer.buffer);
		Sleep(100);
	}

	//追加
	for (auto obj : g_ObjectList)
		delete obj;
	g_ObjectList.clear();

	return 0;
}

/*
* Object指向だとEnemyが居た場所にPlayerを生成して、Enemyを削除と面倒
* コンポーネント指向だとEnemyコンポーネントを削除してPlayerコンポーネントを追加するだけ
*/
```

ここまで学習した事を参考に、prob9_10_11_12 を解け。