Games104_homework4_report

Online version: https://nbqmlgi3yg.feishu.cn/docx/doxcnMZQWujiCLv7Ngm3Yjkk9yg

The content in the online version may be more updated.

Author: huandzh@gmail.com (huandzh (DHuan) (github.com))

Post Date: 2022-08-30

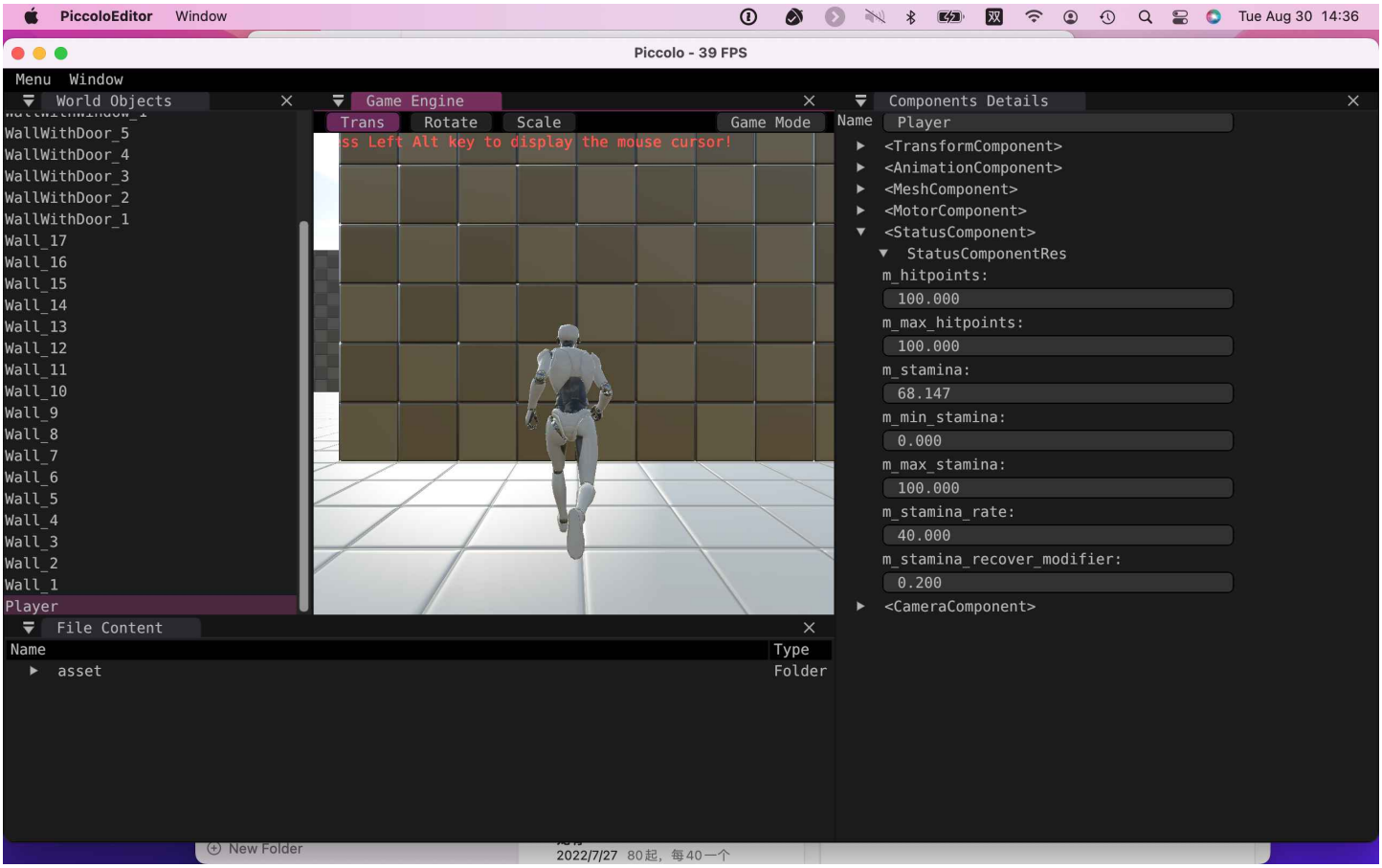
新增或修改属性的定义及意义说明

典型的动作游戏中角色一般拥有生命、体力等属性，为了实现相关的功能增加了`StatusComponent`。

- m_hitpoints - 当前生命值
- m_max_hitpoints - 最大生命值
- m_stamina - 体力值
- m_min_stamina - 最小体力值
- m_max_stamina - 最大体力值
- m_stamina_rate - 体力值消耗速度
- m_stamina_recover_modifier - 体力值回复倍数（可回复体力值时，回复速度是消耗速度的多少倍）

新增或修改属性在 Components Details 面板上显示的截图

`StatusComponent` 挂载在角色游戏对象下和 `MotorComponent` 同级：



使该属性在其系统内生效的代码说明，包括代码解释及实现思路说明

预期效果

作业希望实现：

1. 角色在起跳时消耗体力
2. 其他状态回复体力
3. 当体力小于允许的最小的体力值时，角色将不能再起跳

实现思路

为简化实现，角色唯一消耗体力的状态是角色起跳的上升段，即 `JumpState::rising`。

为此先新增了 `MotorComponent::getIsJumpingUp()` 使得其他的组件方便访问这个状态。

接下来参照 `MotorComponent` 的写法：

- 新增 `StatusComponentRes` 用于定义属性
- 新增 `StatusComponent` 来包括该属性资源，提供每个tick的更新机制，提供体力值耗尽状态的相关方法
- 在 `player.object.json` 中新增相应的初始值

最后，在 `MotorComponent` 和 `StatusComponent` 中实现相应的跳跃和体力值控制代码。

代码及解释：

Piccolo/engine/source/runtime/resource/res_type/components/status.h

定义角色相关的属性，使用tags `Fields` ，这样所有的类属性均会被反射

```
#pragma once

#include "runtime/core/meta/reflection/reflection.h"

namespace Piccolo
{
    REFLECTION_TYPE(StatusComponentRes)
    CLASS(StatusComponentRes, Fields)
    {
        REFLECTION_BODY(StatusComponentRes);

    public:
        StatusComponentRes() = default;

        float m_hitpoints {0.F};
        float m_max_hitpoints {0.F};

        float m_stamina {0.F};
        float m_min_stamina {0.F};
        float m_max_stamina {0.F};
        float m_stamina_rate {0.F};
        float m_stamina_recover_modifier {0.F};
    };
} // namespace Piccolo

```

Piccolo/engine/source/runtime/function/framework/component/status/status_component.h

使用tags `WhiteListFields`，除了明确标上 `Meta(Enable)` 的 `StatusComponentRes` 会被反射外，其他属性均为普通属性，用于实现游戏机制。

定义 `StatusComponent`，包括以下方法：

- `postLoadResource` - 提供父对象
- `tick` - 每帧被自动调用，这里将调用 `tickPlayerStatus`
- `tickPlayerStatus` - 和角色状态有关的机制，这里只实现了部分体力值相关的功能
- `getIsOutOfStamina` - 角色是否已经耗尽体力

```
#pragma once

#include "runtime/resource/res_type/components/status.h"

#include "runtime/function/framework/component/component.h"

namespace Piccolo
{
    REFLECTION_TYPE(StatusComponent)
    CLASS(StatusComponent : public Component, WhiteListFields)
    {
        REFLECTION_BODY(StatusComponent)
    public:
        StatusComponent() = default;

        void postLoadResource(std::weak_ptr<GObject> parent_object) override;

        void tick(float delta_time) override;
        void tickPlayerStatus(float delta_time);
        bool getIsOutOfStamina() const { return m_is_out_of_stamina; }

    private:
        META(Enable)
        StatusComponentRes m_status_res;
        bool               m_is_out_of_stamina {false};
    };
} // namespace Piccolo


```

Piccolo/engine/source/runtime/function/framework/component/status/status_component.cpp

主要实现 `StatusComponent::tickPlayerStatus` 中的游戏机制

- 调用 `m_parent_object.lock()->tryGetComponent` 获取 `MotorComponent` 的跳跃状态
- 如果在跳跃，消耗体力值
- 如果是其他状态，回复体力值，直至回复到最大体力值
- 更新体力值是否被耗尽到 `m_is_out_of_stamina` ， `getIsOutOfStamina()` 将用于获取它的值

以上数值使用被反射的 `StatusComponentRes` 里相关的体力值、最大体力值、体力值消耗速度等属性定义。

```
1  #include "runtime/function/framework/component/status/status_component.h"
2
3  #include "runtime/core/base/macro.h"
4
5  #include "runtime/function/character/character.h"
6  #include "runtime/function/controller/character_controller.h"
7  #include "runtime/function/framework/component/motor/motor_component.h"
8  #include "runtime/function/framework/level/level.h"
9  #include "runtime/function/framework/object/object.h"
10 #include "runtime/function/framework/world/world_manager.h"
11 #include "runtime/function/global/global_context.h"
12 #include "runtime/function/input/input_system.h"
13 #include "runtime/function/physics/physics_scene.h"
14 #include <algorithm>
15
16 namespace Piccolo
17 {
18     void StatusComponent::postLoadResource(std::weak_ptr<GObject> parent_object)
   { m_parent_object = parent_object; }
19
20     void StatusComponent::tick(float delta_time) { tickPlayerStatus(delta_time);
   }
21
22     void StatusComponent::tickPlayerStatus(float delta_time)
23     {
24
25         if (!m_parent_object.lock())
```

```
            return;

        std::shared_ptr<Level> current_level = g_runtime_global_context.m_world_manager->getCurrentActiveLevel().lock();
        std::shared_ptr<Character> current_character = current_level->getCurrentActiveCharacter().lock();
        if (current_character == nullptr)
            return;

        if (current_character->getObjectID() != m_parent_object.lock()->getID())
            return;

        MotorComponent* motor_component = m_parent_object.lock()->tryGetComponent<MotorComponent>("MotorComponent");

        //  jump up consume stamina
        if (motor_component->getIsJumpingUp())
        {
            m_status_res.m_stamina -= m_status_res.m_stamina_rate * delta_time;
        }
        // else recover to max
        else if (m_status_res.m_stamina < m_status_res.m_max_stamina)
        {
            m_status_res.m_stamina = std::min(m_status_res.m_stamina + m_status_res.m_stamina_rate * delta_time *
                                                                       m_status_res.m_stamina_recover_modifier,
                                              m_status_res.m_max_stamina);
        }

        m_is_out_of_stamina = m_status_res.m_stamina < m_status_res.m_min_stamina;
    }

} // namespace Piccolo


```

Piccolo/engine/source/runtime/function/framework/component/status/motor_component.cpp

获取`StatusComponent::getIsOutOfStamina()`，如果耗尽了不再触发跳跃相关机制。

```
    // 代码节选
    void MotorComponent::calculatedDesiredVerticalMoveSpeed(unsigned int command, float delta_time)
```

```
    {
        // ...
        if (m_jump_state == JumpState::idle)
        {
            const StatusComponent* status_component = m_parent_object.lock()->tryGetComponentConst(StatusComponent);

            const bool is_out_of_stamina = status_component->getIsOutOfStamina();
            if (((unsigned int)GameCommand::jump & command) && !is_out_of_stamina)
            {
                m_jump_state                  = JumpState::rising;
                m_vertical_move_speed         = Math::sqrt(m_motor_res.m_jump_height * 2 * gravity);
                m_jump_horizontal_speed_ratio = m_move_speed_ratio;
            }
            else
            {
                m_vertical_move_speed = 0.f;
            }
        }
        else if (m_jump_state == JumpState::rising || m_jump_state == JumpState::falling)
        {
            m_vertical_move_speed -= gravity * delta_time;
            if (m_vertical_move_speed <= 0.f)
            {
                m_jump_state = JumpState::falling;
            }
        }
    }


```

Piccolo/engine/asset/objects/character/player/player.object.json

参照 `MotorComponent` 的配置写法，配置相关的属性值

```
// 节选
        {
            "$context": {
                "status_res": {
                    "hitpoints": 100.0,
                    "max_hitpoints": 100.0,
                    "stamina": 100.0,
                    "max_stamina": 100.0,
```

```
                    "min_stamina": 0.0,
                    "stamina_rate": 40.0,
                    "stamina_recover_modifier": 0.2
                }
            },
            "$typeName": "StatusComponent"
        }

```

实现效果视频

见Games104_homework4_video.mp4。