Moon Engine

Introducción

Moon es un Game Engine desarrollado por KEGE Studios escrito en C/C++, fue diseñado basado en la Arquitectura ECS y usando algunas funcionalidades del nuevo estándar de C++20.

Características principales

- Eficiente en el uso de Recursos
- Arquitectura Kernel para el escalamiento e interacción entre los plugins
- Uso de la STL para los contenedores
- Concepts implementados para una mayor seguridad en templates
- Predicción de tipos en tiempo de compilación
- Un Core simple, liviano y libre
- Plugins pequeños y específicos
- Ejemplos para todos los plugins oficiales
- Solo incluyes lo que necesitas
- Independencia de plugins

Caracteristicas de la versión

- GameContext
- System
- Component
- Entity
- Herramienta de Instalacion y creacion de proyectos
- Uso de la STL para los contenedores
- Predicción de tipos en tiempo de compilación

Configuración del entorno

Añade la variable de entorno

```
MOON_STUDIO_PATH=$HOME/MoonStudio
```

Esto servirá para la creación de los proyectos de Moon

Instalación

Plataformas

- Windows
- Linux
- MacOS(Próximamente)

Windows y Linux

Solo se requiere tener instalado CMake y algún Ninja junto a un compilador de C++ de su elección se recomienda g++ de cygwin para Windows y GNU GCC o Clang para Linux.

```
mkdir MoonStudio
cd MoonStudio
git clone git@github.com:reitmas32/Moon.git
cd Moon
mkdir build
cd build
cmake .. -G <BuildSystem>-DCMAKE_CXX_COMPILER=<CXX_COMPILER> -DCMAKE_C_COMPILER=<C_COMPIL
```

Hola Mundo

Primero necesitamos crear el directorio de nuestro proyecto

```
mkdir hello-world
cd hello-world
```

Ahora crearemos algunas carpetas para organizar el proyecto

```
mkdir src/ include/ cmake/
```

Para la compilacion de nuestro proyecto aligual que hicimos con Moon usaremos CMake, los siguientes archivos serviran para la configuración

```
# CMakeLists.txt

########################
#     Config  of       #
```

```
#       Project        #
#########################
cmake_minimum_required (VERSION 3.0.0)

set(NAME_PROJECT HelloMoon)

set(APP hello-moon)

if (CMAKE_VERSION VERSION_LESS 3.0)
    PROJECT(${NAME_PROJECT} CXX)
else()
    cmake_policy(SET CMP0048 NEW)
    PROJECT(${NAME_PROJECT} VERSION "1.0.0" LANGUAGES CXX)
endif()

if(${WIN32})
    if(EXISTS ${CMAKE_CURRENT_SOURCE_DIR}/cmake/windows.cmake)
        include(${CMAKE_CURRENT_SOURCE_DIR}/cmake/windows.cmake)
    endif()
elseif(${UNIX})
    set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++17")
    if(EXISTS ${CMAKE_CURRENT_SOURCE_DIR}/cmake/linux.cmake)
        include(${CMAKE_CURRENT_SOURCE_DIR}/cmake/linux.cmake)
    endif()
endif()



#   cmake/linux.cmake

#########################
#     Main Dirs of      #
#       Project         #
#########################
set(SOURCE_DIR ${CMAKE_CURRENT_SOURCE_DIR}/src)
set(INCLUDE_DIR ${CMAKE_CURRENT_SOURCE_DIR}/include)
set(BUILD_DIR ${CMAKE_CURRENT_SOURCE_DIR}/build)

#Add Dependencies
include(cmake/vendor.cmake)

#########################
#    Include Dirs of    #
#       Project         #
#########################
include_directories(
    .
    ${INCLUDE_DIR}
    #Includes of Mooon
    ${MOON_INCLUDE_DIR}
)
```

```
########################
#    Find Source of    #
#       Project        #
########################
file( GLOB_RECURSE LIB_SOURCES ${SOURCE_DIR}/*.cpp )
file( GLOB_RECURSE LIB_HEADERS ${INCLUDE_DIR}/*.hpp )

########################
#     Add Source of    #
#       Project        #
########################
#TODO: add_library
add_executable(
    ${APP}
    ${LIB_SOURCES}
    ${LIB_HEADERS}
)
target_link_libraries(${APP}
    ${MOON_LIBRARIES}
    )



#   cmake/windows.cmake

########################
#     Main Dirs of     #
#       Project        #
########################
set(SOURCE_DIR ${CMAKE_CURRENT_SOURCE_DIR}/src)
set(INCLUDE_DIR ${CMAKE_CURRENT_SOURCE_DIR}/include)
set(BUILD_DIR ${CMAKE_CURRENT_SOURCE_DIR}/build)

#Add Dependencies
include(cmake/vendor.cmake)

########################
#    Include Dirs of   #
#       Project        #
########################
include_directories(
    .
    ${INCLUDE_DIR}
    #Includes of Mooon
    ${MOON_INCLUDE_DIR}
)

########################
#    Find Source of    #
#       Project        #
########################
file( GLOB_RECURSE LIB_SOURCES ${SOURCE_DIR}/*.cpp )
```

```
file( GLOB_RECURSE LIB_HEADERS ${INCLUDE_DIR}/*.hpp )

########################
#     Add Source of    #
#       Project        #
########################
#TODO: add_library
add_executable(
    ${APP}
    ${LIB_SOURCES}
    ${LIB_HEADERS}
)
target_link_libraries(${APP}
    ${MOON_LIBRARIES}
    )



#   cmake/vendor.cmake

########################
#     Include Moon     #
########################
include(cmake/moon.cmake)



# cmake/moon.cmake

########################
#       Lib Moon       #
########################
if(${WIN32})
    #   Dir of MoonStudio
    set(MOON_STUDIO_DIR $ENV{MOON_STUDIO_PATH})

    set(MOON_INCLUDE_DIR
        ${MOON_STUDIO_DIR}
        ${MOON_STUDIO_DIR}/Moon/template
        ${MOON_STUDIO_DIR}/Moon/include
        ${MOON_STUDIO_DIR}/Moon/vendor/termcolor)

    message(${MOON_INCLUDE_DIR})
    option(LOGS "ON")
    if(${LOGS} STREQUAL "OFF")
        #add_compile_definitions(RELEASE)
        set(MOON_LIBRARIES ${MOON_STUDIO_DIR}/Moon/build/libmoon.a)
    else()
        set(MOON_LIBRARIES ${MOON_STUDIO_DIR}/Moon/build/libmoon.a)
    endif()
```

```
elseif(${UNIX})
    #   Dir of MoonStudio
    set(MOON_STUDIO_DIR $ENV{MOON_STUDIO_PATH})

    set(MOON_INCLUDE_DIR
        ${MOON_STUDIO_DIR}
        ${MOON_STUDIO_DIR}/Moon/template
        ${MOON_STUDIO_DIR}/Moon/include
        ${MOON_STUDIO_DIR}/Moon/vendor/termcolor)

    message(${MOON_INCLUDE_DIR})
    option(LOGS "ON")
    if(${LOGS} STREQUAL "OFF")
        #add_compile_definitions(RELEASE)
        set(MOON_LIBRARIES ${MOON_STUDIO_DIR}/Moon/build/libmoon.a)
    else()
        set(MOON_LIBRARIES ${MOON_STUDIO_DIR}/Moon/build/libmoon.a)
    endif()

endif()
```

Despues de esta configuaración creamos un archivo main.cpp y creamos nuestro primer component

```
//  src/main.cpp

#include <iostream>

//Info Platform
#include <Moon/include/tools/platform_info.hpp>

//Component
#include <Moon/template/core/cmp/cmp.tpp>

//MyComponent
struct MyCmp_t : Moon::Core::Component_t<MyCmp_t>
{
    //data of Component
    int data;

    //Constructors
    MyCmp_t(Moon::Alias::EntityId eid) : Moon::Core::Component_t<MyCmp_t>(eid) {}
    MyCmp_t(
        Moon::Alias::EntityId eid,
        int data) : Moon::Core::Component_t<MyCmp_t>(eid), data{data}{}
    MyCmp_t() = default;

    //Destructors
    ~MyCmp_t() = default;
};
```

```
int main(int argc, char const *argv[])
{
    MyCmp_t cmp = MyCmp_t(0, 45); // eid = 0, data = 45
    std::cout << "MyCmp_t eid: "<<cmp.eid<<", data: " << cmp.data << std::endl;
    return 0;
}
```

ECS(Entity Component System)

La manipulación de los datos de un vídeo juego es una tarea compleja de administrar cuando uno esta iniciando en el desarrollo de vídeo juegos, podrías terminar con un acoplamiento entre las partes que lo componen a tal punto en el que el modulo encargado de la física también manipule datos de renderizado. Por esta razón es importante tener una buena planificación.

La arquitectura ECS "Entity Component System" es una forma sencilla y eficiente de administrar los recursos de un vídeo juego, como su nombre lo indica esta se basa en separa los Datos "Entity Component" de la manipulación e interpretación de los mismos "System"

Moon Docs

ComponentBase_t

Clase para crear CRTP de Component_t

Definición

```
/**
 * @file Moon/include/core/cmp/ccmp_base.hpp
 */
struct ComponentBase_t
```

Metodos públicos
ComponentBase_t()
~ComponentBase_t()

Miembros públicos estaticos
Moon::Alias::ComponentType nextType = 0

Miembros públicos
Moon::Alias::EntityId eid = 0

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **ComponentBase_t()**
  Constructor por defecto
  Definicion

```
Moon::Core::ComponentBase_t::ComponentBase_t()
```

- **~ComponentBase_t()**
  Destructor por defecto
  Definicion

```
Moon::Core::ComponentBase_t::~ComponentBase_t()
```

Component_t

Clase de la que heredan todos los compnentes

Los Components son una de las partes mas importantes de la arquitectura ECS para videojuegos, con estos se dota de caracteristicas a las Entidades para que los sistemas puedan modificar el comportamiento de estas.

Herencia

ComponentBase_t

Definición

```
/**
 * @file Moon/include/core/cmp/cmp.hpp
 */
template \<class Type>
struct Component_t : public ComponentBase_t
```

Metodos públicos
Component_t ()
~Component_t ()
Component_t (Moon::Alias::EntityId eid)

Metodos públicos estaticos
static Moon::Alias::ComponentType getComponentType ()

Miembros heredados
Moon::Alias::EntityId eid

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Ejemplo

```
struct Sprite_t : Moon::Core::Component_t<Sprite_t>{
private:
     std::vector<Image> sprite;
public:
     Sprite_t(){}

     Sprite_t(Moon::Alias::EntityId eid) : Component_t<Sprite_t>(eid){
         //TODO:All Constructor
     }

     ~Sprite_t(){
         //TODO:All Destructor
     }
```

```
};
```

Contructores y Destructores

- **Component_t() [1/2]**
  Constructor por defecto
  Definicion

```
template<class Type>
Moon::Core::Component_t<Type>::Component_t()
```

- **Component_t(Moon::Alias::EntityId eid) [2/2]**
  Constructor con inyeccion de Moon::Alias::EntityId
  Definicion

```
template<class Type>
Moon::Core::Component_t<Type>::Component_t(Moon::Alias::EntityId eid)
```

- **~Component_t()**
  Destructor por defecto
  Definicion

```
template<class Type >
virtual Moon::Core::Component_t< Type >::~Component_t()
```

Funciones Miembro

- **getComponentType()** Obtiene el Tipo de Componente

```
/**
* @pre None
* @post The return is unique for each Component_t
* @return Moon::Alias::ComponentType
*/
template<class Type >
static Moon::Alias::ComponentType Moon::Core::Component_t< Type >::getComponentType
```

ComponentSingleton_t

Clase de la que heredan todos los compnentes *singleton*

Los Components son una de las partes mas importantes de la arquitectura ECS para videojuegos, con estos se dota de caracteristicas a las Entidades para que los sistemas puedan modificar el comportamiento de estas.

Herencia

ComponentBase_t

Definición

```
/**
 * @file Moon/include/core/cmp/cmp.hpp
 */
template \<class Type>
struct ComponentSingleton_t : protected ComponentBase_t
```

Metodos públicos
ComponentBase_t ()
~ComponentBase_t ()

Metodos públicos estaticos
static Moon::Alias::ComponentType getComponentType ()

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Ejemplo

```
struct MediaQuery_t : Moon::Core::ComponentSingleton_t<MediaQuery_t>{
private:
     Size size;
     bool fullScreen;
public:
     MediaQuery_t(){}
```

```
        ~MediaQuery_t(){
            //TODO:All Destructor
        }
};
```

Contructores y Destructores

- **ComponentSingleton_t()**
  Constructor por defecto
  Definicion

```
template<class Type>
Moon::Core::ComponentSingleton_t<Type>::ComponentSingleton_t()
```

- **~ComponentSingleton_t()**
  Destructor por defecto
  Definicion

```
template<class Type >
virtual Moon::Core::ComponentSingleton_t< Type >::~ComponentSingleton_t()
```

Funciones Miembro

- **getComponentType()** Obtiene el Tipo de Componente

```
/**
* @pre None
* @post The return is unique for each Component_t
* @return Moon::Alias::ComponentType
*/
template<class Type >
static Moon::Alias::ComponentType Moon::Core::ComponentSingleton_t< Type >::getComp
```

ComponentBaseVect_t

Abstracción de un ComponentVect_t

Definición

```
/**
 * @file Moon/include/core/cmp/cmp_vect.hpp
 */
struct ComponentBaseVect_t
```

Metodos públicos
ComponentBaseVect_t()
~ComponentBaseVect_t()
deleteComponentByEntityId(Moon::Alias::EntityId eid)

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **ComponentBaseVect_t()**
  Constructor por defecto
  Definicion

```
Moon::Core::ComponentBaseVect_t::ComponentBaseVect_t()
```

- **~ComponentBaseVect_t()**
  Destructor por defecto
  Definicion

```
virtual Moon::Core::ComponentBaseVect_t::~ComponentBaseVect_t()
```

Funciones Miembro

- **deleteComponentByEntityId(Moon::Alias::EntityId eid)** Elimina un Component_t usando eid

```
/**
* @param eid Id of the Entity to which the Component belongs
* @return ComponentBase_t* pointer of Component_t Delete
```

```
     */
     virtual Moon::Core::ComponentBase_t *Moon::Core::ComponentBaseVect_t::deleteComponent
```

ComponentVect_t

Wrapper de un vector de Component_t<Type>

Herencia

ComponentBaseVect_t

Definición

```
/**
 * @file Moon/include/core/cmp/cmp_vect.hpp
 */
template <MOON_IS_CMP_T CMP_t>
struct ComponentVect_t : ComponentBaseVect_t
```

Metodos públicos
ComponentVect_t()
~ComponentVect_t()
findComponentIteratorById(Moon::Alias::EntityId eid)
deleteComponentByEntityId(Moon::Alias::EntityId eid)

Metodos públicos heredados
deleteComponentByEntityId(Moon::Alias::EntityId eid)

Miembros públicos
Moon::Alias::EntityId eidstd::vector<CMP_t> components

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **ComponentVect_t()**
  Constructor por defecto
  Definicion

```
template<MOON_IS_CMP_T CMP_t>
Moon::Core::ComponentVect_t<CMP_t>::ComponentVect_t()
```

- **~ComponentVect_t()**
  Destructor por defecto
  Definicion

```
template<MOON_IS_CMP_T CMP_t >
virtual Moon::Core::ComponentVect_t< CMP_t >::~ComponentVect_t()
```

Funciones Miembro

- **findComponentIteratorById(Moon::Alias::EntityId eid)** Busca un Componente utilizando *eid*

```
/**
* @param eid Id of the Entity to which the Component belongs
* @return Component_t* with Id equal eid
*/
template<MOON_IS_CMP_T CMP_t >
constexpr auto Moon::Core::ComponentVect_t< CMP_t >::findComponentIteratorById(Moon
```

- **deleteComponentByEntityId(Moon::Alias::EntityId eid)** Elimina un Componente utilizando *eid*

```
/**
* @param eid Id of the Entity to which the Component belongs
* @return ComponentBase_t* pointer of Component_t Delete
*/
template<MOON_IS_CMP_T CMP_t >
Moon::Core::ComponentBase_t *Moon::Core::ComponentVect_t< CMP_t >::deleteComponentB
```

ComponentStorage_t

Clase Contenedora de ComponentVect_t<Type>

Definición

```
/**
 * @file Moon/include/core/cmp/cmp_storage.hpp
 */
struct ComponentStorage_t
```

Metodos públicos
ComponentStorage_t()
~ComponentStorage_t()
createComponent(Moon::Alias::EntityId eid, Ts &&...args)
getComponents()
getComponents() const
deleteComponentByTypeIdAndEntityId(Moon::Alias::ComponentType cid, Moon::Alias::EntityId eid)

Metodos privados
createComponentVector()

Miembros privados
std::unordered_map<Moon::Alias::ComponentType, std::unique_ptr<ComponentBaseVect_t>> storage

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **ComponentStorage_t()** Constructo por defecto
  Definición

```
struct Moon::Core::ComponentStorage_t::ComponentStorage_t()
```

- **~ComponentStorage_t()** Destructor por defecto
  Definición

```
struct Moon::Core::ComponentStorage_t::~ComponentStorage_t()
```

Funciones Miembro

- **createComponent(Moon::Alias::EntityId eid, Ts &&...args)** Crea un Component_t y lo almacena en el Contenedor Definicion

```
/**
* @tparam CMP_t Type of new Component_t
* @param eid Id of the Entity to which the Component belongs
* @param Ts... Params of Contructor of Component_t
* @pre The Component_t to create must have a constructor with the specified parameter
* @return CMP_t& This is a reference to new Component_t
*/
template <MOON_IS_CMP_T CMP_t, typename... Ts>
CMP_t &Moon::Core::ComponentStorage_t::createComponent(Moon::Alias::EntityId eid, Ts
```

- **getComponents()** Obtiene el vector del tipo de Component_t solicitado Definicion

```
/**
* @tparam CMP_t Type of the Component_t
* @return std::vector<CMP_t>& Reference to vector of Component_t's
*/
template <MOON_IS_CMP_T CMP_t>
std::vector<CMP_t> &Moon::Core::ComponentStorage_t::getComponents()
```

- **getComponents() const** Obtiene el vector del tipo de Component_t solicitado Definicion

```
/**
* @tparam CMP_t Type of the Component_t
* @return std::vector<CMP_t>& Reference to vector of Component_t's
*/
template <MOON_IS_CMP_T CMP_t>
const std::vector<CMP_t> &Moon::Core::ComponentStorage_t::getComponents() const
```

- **deleteComponentByTypeIdAndEntityId(Moon::Alias::ComponentType cid, Moon::Alias::EntityId eid)** Elimina un Component_t del tipo seleccionado y el eid Definicion

```
/**
* @param cid Id of the Component
* @param eid Id of the Entity to which the Component belongs
* @return ComponentBase_t* pointer of Component_t Delete
*/
ComponentBase_t *
deleteComponentByTypeIdAndEntityId(Moon::Alias::ComponentType cid, Moon::Alias::Entit
```

Funciones Miembro privadas

- **createComponentVector()** Crea un std::vector<CMP_t> del tipo indicado Definicion

```
/**
* @tparam CMP_t Type of the Component_t
* @return std::vector<CMP_t>&
*/
template <MOON_IS_CMP_T CMP_t>
std::vector<CMP_t> &createComponentVector()
```

EntityBase_t

Clase para crear CRTP de Component_t

Definición

```
/**
 * @file Moon/include/core/ent/ent_base.hpp
 */
struct EntityBase_t
```

Metodos públicos
EntityBase_t()
~EntityBase_t()
addComponent(CMP_t *cmp)
getComponent()

Miembros públicos estaticos
Moon::Alias::ComponentType nextType = 0

Miembros públicos
MapCmps_t components

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **EntityBase_t()**
  Constructor por defecto
  Definicion

```
Moon::Core::EntityBase_t::EntityBase_t()
```

- **~Entity_t()**
  Destructor por defecto
  Definicion

```
Moon::Core::EntityBase_t::~EntityBase_t()
```

Funciones Miembro

- **addComponent(CMP_t *cmp)** Añade una referencia al Component_t *cmp* Definicion

```
/**
* @tparam CMP_t Type of the Component_t
* @pre CMP_t is base of Moon::Core::ComponentBase_t
* @param cmp pointer of Component_t
*/
template <MOON_IS_CMP_T CMP_t>
void
Moon::Core::Entity_t::addComponent(CMP_t *cmp)
```

- **getComponent()** Obtiene una referencia al Component_t *cmp* Definicion

```
/**
* @tparam CMP_t
* @pre CMP_t is base of Moon::Core::ComponentBase_t
* @return CMP_t*
*/
template <MOON_IS_CMP_T CMP_t>
CMP_t *Moon::Core::Entity_t::getComponent()
```

Entity_t

Clase de la que heredan todas la entidades

Las Enidades son una parte esencial de la arquitectura ECS, en estas se almacenan un conjunto de Componentes de distintos tipos estos tiene relacion directa entre si los cuales dotan de caracteristicas a las entidades

Herencia

EntityBase_t

Definición

```
/**
 * @file Moon/include/core/ent/entity.hpp
 */
template <class Type>
struct Entity_t : public EntityBase_t
```

Metodos públicos
Entity_t ()
~Entity_t ()
Entity_t (Moon::Alias::EntityId eid)
void updateComponent(Moon::Alias::ComponentType cid, ComponentBase_t *cmp_ptr)
auto begin()
auto end()

Metodos públicos estaticos
static Moon::Alias::EntityType getEntityType ()

Miembros heredados
MapCmps_t components

Miembros públicos
Moon::Alias::EntityId eid = 0

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Ejemplo

```
struct Pacman : Moon::Core::Entity_t<Pacman>{
public:
     Pacman(){}

     Pacman(Moon::Alias::EntityId eid) : Entity_t<Pacman>(eid){
         //TODO:All Constructor
     }

     ~Pacman(){
         //TODO:All Destructor
     }
};
```

Contructores y Destructores

- **Entity_t() [1/2]**
  Constructor por defecto
  Definicion

```
template<class Type>
Moon::Core::Entity_t<Type>::Entity_t()
```

- **Entity_t(Moon::Alias::EntityId eid) [2/2]**
  Constructor con inyeccion de Moon::Alias::EntityId
  Definicion

```
template<class Type>
Moon::Core::Entity_t<Type>::Entity_t(Moon::Alias::EntityId eid)
```

- **~Entity_t()**
  Destructor por defecto
  Definicion

```
template<class Type >
virtual Moon::Core::Entity_t< Type >::~Entity_t()
```

Funciones Miembro

- **getComponentType()** Obtiene el Tipo de Componente

```
/**
* @pre None
* @post The return is unique for each Entity_t
* @return Moon::Alias::EntityType
*/
template<class Type >
static Moon::Alias::EntityType Moon::Core::Entity_t< Type >::getEntityType()
```

- **updateComponent(Moon::Alias::ComponentType cid, ComponentBase_t *cmp_ptr)** Actualiza la referenica a un Component_t

```
/**
* @pre cmp_ptr not nullptr
* @post The return is unique for each Entity_t
* @return Moon::Alias::EntityType
*/
template<class Type >
void Moon::Core::Entity_t< Type >::updateComponent(Moon::Alias::ComponentType cid,
```

- **begin()** Genera el mixin begin() para que Entity_t sea iterable

```
/**
* @post The return is Map for Component_t
```

```
        * @return MapCmps_t
        */
        template<class Type >
        auto Moon::Core::Entity_t< Type >::begin()
```

- **end()** Genera el mixin end() para que Entity_t sea iterable

```
        /**
        * @post The return is Map for Component_t
        * @return MapCmps_t
        */
        template<class Type >
        auto Moon::Core::Entity_t< Type >::end()
```

EntityBaseVect_t

Abstracción de un EntityVect_t

Definición

```
/**
 * @file Moon/include/core/ent/ent_vect.hpp
 */
struct EntityBaseVect_t
```

Metodos públicos
EntityBaseVect_t()
~EntityBaseVect_t()
deleteEntityByEntityId(Moon::Alias::EntityId eid)

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **EntityBaseVect_t()**
  Constructor por defecto
  Definicion

```
Moon::Core::EntityBaseVect_t::EntityBaseVect_t()
```

- **~EntityBaseVect_t()**
  Destructor por defecto
  Definicion

```
virtual Moon::Core::EntityBaseVect_t::~EntityBaseVect_t()
```

Funciones Miembro

- **deleteEntityByEntityId(Moon::Alias::EntityId eid)** Elimina un Entity_t usando eid

```
/**
* @param eid Id of the Entity to which the Entity belongs
* @return EntityBase_t* pointer of Entity_t Delete
*/
virtual Moon::Core::EntityBase_t *Moon::Core::EntityBaseVect_t::deleteEntityByEntityI
```

EntityVect_t

Wrapper de un vector de Entity_t<Type>

Herencia

EntityBaseVect_t

Definición

```
/**
 * @file Moon/include/core/ent/ent_vect.hpp
 */
template <MOON_IS_ENT_T ENT_t>
struct EntityVect_t : EntityBaseVect_t
```

Metodos públicos

Metodos públicos
EntityVect_t()
~EntityVect_t()
findEntityIteratorById(Moon::Alias::EntityId eid)
deleteEntityByEntityId(Moon::Alias::EntityId eid)

Metodos públicos heredados
deleteEntityByEntityId(Moon::Alias::EntityId eid)

Miembros públicos
Moon::Alias::EntityId eid
std::vector<ENT_t> entities

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **EntityVect_t()**
  Constructor por defecto
  Definicion

```
template<MOON_IS_ENT_T ENT_t>
Moon::Core::EntityVect_t<ENT_t>::EntityVect_t()
```

- **~EntityVect_t()**
  Destructor por defecto
  Definicion

```
template<MOON_IS_ENT_T ENT_t>
virtual Moon::Core::EntityVect_t< ENT_t >::~EntityVect_t()
```

Funciones Miembro

- **findEntityIteratorById(Moon::Alias::EntityId eid)** Busca un Entity utilizando *eid*

```
/**
* @param eid Id of the Entity to which the Entity belongs
* @return Entity_t* with Id equal eid
*/
template<MOON_IS_ENT_T ENT_t>
constexpr auto Moon::Core::EntityVect_t< ENT_t >::findEntityIteratorById(Moon::Alia
```

- **deleteEntityByEntityId(Moon::Alias::EntityId eid)** ELimina un Entity utilizando *eid*

```
/**
* @param eid Id of the Entity to which the Entity belongs
* @return EntityBase_t* pointer of Entity_t Delete
*/
template<MOON_IS_ENT_T ENT_t>
Moon::Core::EntityBase_t *Moon::Core::EntityVect_t< ENT_t >::deleteEntityByEntityId
```

EntityStorage_t

Clase Contenedora de EntityVect_t<Type>

Definición

```
/**
 * @file Moon/include/core/ent/ent_storage.hpp
 */
struct EntityStorage_t
```

Metodos públicos
EntityStorage_t()
~EntityStorage_t()
createEntity(Moon::Alias::EntityId eid, Ts &&...args)
getEntities()
getEntities() const

Metodos públicos
deleteEntityByTypeIdAndEntityId(Moon::Alias::EntityType cid, Moon::Alias::EntityId eid)

Metodos privados
createEntityVector()

Miembros privados
std::unordered_map<Moon::Alias::EntityType, std::unique_ptr<EntityBaseVect_t>> storage

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **EntityStorage_t()** Constructo por defecto
  Definición

```
struct Moon::Core::EntityStorage_t::EntityStorage_t()
```

- **~EntityStorage_t()** Destructor por defecto
  Definición

```
struct Moon::Core::EntityStorage_t::~EntityStorage_t()
```

Funciones Miembro

- **createEntity(Moon::Alias::EntityId eid, Ts &&...args)** Crea un Entity_t y lo almacena en el Contenedor Definicion

```
/**
* @tparam ENT_t Type of new Entity_t
* @param eid Id of the Entity to which the Entity belongs
* @param Ts... Params of Contructor of Entity_t
* @pre The Entity_t to create must have a constructor with the specified parameters
```

```
* @return ENT_t& This is a reference to new Entity_t
*/
template <MOON_IS_ENT_T ENT_t, typename... Ts>
ENT_t &Moon::Core::EntityStorage_t::createEntity(Moon::Alias::EntityId eid, Ts &&...a
```

- **getEntities()** Obtiene el vector del tipo de Entity_t solicitado Definicion

```
/**
* @tparam ENT_t Type of the Entity_t
* @return std::vector<ENT_t>& Reference to vector of Entity_t's
*/
template <MOON_IS_ENT_T ENT_t>
std::vector<ENT_t> &Moon::Core::EntityStorage_t::getEntities()
```

- **getEntities() const** Obtiene el vector del tipo de Entity_t solicitado Definicion

```
/**
* @tparam ENT_t Type of the Entity_t
* @return std::vector<ENT_t>& Reference to vector of Entity_t's
*/
template <MOON_IS_ENT_T ENT_t>
const std::vector<ENT_t> &Moon::Core::EntityStorage_t::getEntities() const
```

- **deleteEntityByTypeIdAndEntityId(Moon::Alias::EntityType cid, Moon::Alias::EntityId eid)** Elimina un Entity_t del tipo seleccionado y el eid Definicion

```
/**
* @param cid Id of the Entity
* @param eid Id of the Entity to which the Entity belongs
* @return EntityBase_t* pointer of Entity_t Delete
*/
EntityBase_t *
deleteEntityByTypeIdAndEntityId(Moon::Alias::EntityType cid, Moon::Alias::EntityId ei
```

Funciones Miembro privadas

- **createEntityVector()** Crea un std::vector<ENT_t> del tipo indicado Definicion

```
/**
* @tparam ENT_t Type of the Entity_t
* @return std::vector<ENT_t>&
*/
```

```
    template <MOON_IS_ENT_T ENT_t>
    std::vector<ENT_t> &createEntityVector()
```

GameContextBase_t

Clase para crear CRTP de GameContext_t

Cuando realizamos un vídeo juego debemos encontrar la manera de almacenar los datos de las entidades de tal forma que los sistemas que modifican éstos datos puedan acceder a ellos de la forma mas rápida posible.

Definición

```
/**
 * @file Moon/include/core/gtx/gtx_base.hpp
 */
struct GameContextBase_t
```

Metodos públicos
GameContextBase_t ()
~GameContextBase_t ()

Miembros publicos estaticos
Moon::Alias::EntityId nextId{ 0 }
Moon::Alias::GameContextType nextType{ 0 }

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **GameContextBase_t()**
  Constructor por defecto

Definicion

```
Moon::Core::GameContextBase_t()
```

- **~GameContextBase_t()**
  Destructor por defecto
  Definicion

```
virtual Moon::Core::~GameContextBase_t()
```

GameContext_t

Las Enidades son una parte esencial de la arquitectura ECS, en estas se almacenan un conjunto de Componentes de distintos tipos estos tiene relacion directa entre si los cuales dotan de caracteristicas a las entidades

Herencia

GameContextBase_t

Definición

```
/**
 * @file Moon/include/core/gtx/gtx.hpp
 */
template <typename Type>
struct GameContext_t : public GameContextBase_t
```

Metodos públicos
GameContext_t ()
~GameContext_t ()
ENT_t &addEntity(Ts &&...args)
ENT_t *getEntityById(Moon::Alias::EntityId eid)
CMP_t &addComponentById(Moon::Alias::EntityId eid, Ts &&...args)
std::vector/<CMP_t> &getComponents()

Metodos públicos
std::vector/<ENT_t> &getEntities()
void destroyEntityById(Moon::Alias::EntityId eid)
CMP_t *getRequiredComponent(Moon::Alias::EntityId eid)

Metodos públicos estaticos
static Moon::Alias::GameContextType getGameContextType ()

Miembros públicos
EntityStorage_t entities
ComponentStorage_t components

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Ejemplo

```
struct Gtx_t : Moon::Core::GameContext_t<Gtx_t>
{
    Gtx_t();
    ~Gtx_t();
};

//Use Gtx_t

//Create
auto gtx = Gtx_t();

//Add Entities
auto ent_1 = gtx.addEntity<Triangle_t>();
auto ent_2 = gtx.addEntity<Triangle_t>();
auto ent_3 = gtx.addEntity<Triangle_t>();

//Add Cmp to ent_1
gtx.addComponentById<Triangle_t, SpriteTriangle_t>(ent_1.eid, 0.2f);
```

```
gtx.addComponentById<Triangle_t, Position2Df_t>(ent_1.eid, 0.0f, 0.0f);
gtx.addComponentById<Triangle_t, PhysicsCmp_t>(ent_1.eid, true);

//Add Cmp to ent_2
gtx.addComponentById<Triangle_t, SpriteTriangle_t>(ent_2.eid, 0.15f);
gtx.addComponentById<Triangle_t, Position2Df_t>(ent_2.eid, 0.9f, -0.9f);
gtx.addComponentById<Triangle_t, IACmp_t>(ent_2.eid);

//Add Cmp to ent_3
gtx.addComponentById<Triangle_t, SpriteTriangle_t>(ent_3.eid, 0.1f);
gtx.addComponentById<Triangle_t, Position2Df_t>(ent_3.eid, -0.5f, -0.5f);
gtx.addComponentById<Triangle_t, RotateCmp_t>(ent_3.eid);

//Game Loop

while(...){
     .
     .
     .
     physicsSys.update(&gtx);
     renderSys.update(&gtx);
     inputSys.update();
     .
     .
     .
}

//In Updates of Systems
struct PhysicsSys_t : Moon::Core::System_t<Gtx_t>{
     void update(Gtx_t *gtx) override{
         for (Position2Df_t &cmp : gtx->getComponents<Position2Df_t>())
          {
              //Any Update cmp Physics
          }
     }
};
```

Contructores y Destructores

- **GameContext_t**
  Constructor por defecto
  Definicion

```
template<typename Type>
Moon::Core::GameContext_t<Type>::GameContext_t()
```

- **~GameContext_t()**
  Destructor por defecto
  Definicion

```
template<typename Type >
virtual Moon::Core::GameContext_t< Type >::~GameContext_t()
```

Funciones Miembro

- **ENT_t &addEntity(Ts &&...args)** Añade una Entity

```
/**
* @tparam ENT_t Type of Entity
* @param args params for the constructor of ENT_t
* @return Type&
*/
template <typename ENT_t, typename... Ts>
ENT_t &Moon::Core::GameContext_t::addEntity(Ts &&...args)
```

- **ENT_t *getEntityById(Moon::Alias::EntityId eid)** Regresa la Entity requerida

```
/**
* @tparam ENT_t Type of Entity
* @param eid
* @return ENT_t* reference to Entity
*/
template <typename ENT_t>
ENT_t *Moon::Core::GameContext_t::getEntityById(Moon::Alias::EntityId eid)
```

- **CMP_t &addComponentById(Moon::Alias::EntityId eid, Ts &&...args)** Añade un Component

```
/**
* @tparam ENT_t Type of Entity
* @tparam CMP_t Type of Component
* @param eid id of Entity
* @param args params for constructor of Component
* @return Cmp_t&
*/

template <typename ENT_t, MOON_IS_CMP_T CMP_t, typename... Ts>
CMP_t &Moon::Core::GameContext_t::addComponentById(Moon::Alias::EntityId eid, Ts &&..
```

- **std::vector<CMP_t> &getComponents()** Regresa los components

```
/**
* @tparam CMP_t Type of components
* @return std::vector<CMP_t>&
*/
template <MOON_IS_CMP_T CMP_t>
std::vector<CMP_t> &Moon::Core::GameContext_t::getComponents()
```

- **std::vector<ENT_t> &getEntities()** Regresa los entities

```
/**
* @tparam ENT_t
* @return std::vector<ENT_t>&
*/
template <typename ENT_t>
std::vector<ENT_t> &Moon::Core::GameContext_t::getEntities()
```

- **void destroyEntityById(Moon::Alias::EntityId eid)** Delete la Entity seleccionada

```
/**
* @tparam ENT_t Type of Entity
* @param eid if of Entity
*/
template <typename ENT_t>
void Moon::Core::GameContext_t::destroyEntityById(Moon::Alias::EntityId eid)
```

- **CMP_t *getRequiredComponent(Moon::Alias::EntityId eid)** Regresa el Component requerido

```
/**
* @tparam CMP_t Type of Component
* @param eid if of Entity
*/
template <MOON_IS_CMP_T CMP_t>
CMP_t *Moon::Core::GameContext_t::getRequiredComponent(Moon::Alias::EntityId eid)
```

SystemBase_t

Clase para crear CRTP de System_t

Definición

```
/**
 * @file Moon/include/core/sys/sys_base.hpp
```

```
 */
struct SystemBase_t
```

Metodos públicos
SystemBase_t ()
~SystemBase_t ()

Metodos públicos virtuales
virtual bool alive()

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Contructores y Destructores

- **SystemBase_t()**
  Constructor por defecto
  Definicion

```
Moon::Core::SystemBase_t()
```

- **~SystemBase_t()**
  Destructor por defecto
  Definicion

```
virtual Moon::Core::~SystemBase_t()
```

Funciones Miembro

- **bool alive() = 0** Indica si el System_t esta activo

```
bool alive() = 0
```

System_t

La manipulación de los datos de un vídeo juego es una tarea compleja de administrar cuando uno esta iniciando en el desarrollo de vídeo juegos, podrías terminar con un acoplamiento entre las partes que le componen a tal punto en el que el modulo encargado de la física también manipule datos de renderizado. Por esta razón es importante tener una buena planificación. La arquitectura ECS "Entity Component System" es una forma sencilla y eficiente de administrar los recursos de un vídeo juego, como su nombre lo indica esta se basa en separa los Datos "Entity Component" de la manipulación e interpretación de los mismos "System"

Herencia

SystemBase_t

Definición

```
/**
 * @file Moon/include/core/sys/sys.hpp
 */
template <MOON_IS_CTX_T... Type>
struct System_t : public SystemBase_t
```

Metodos públicos
System_t ()
~System_t ()
virtual void update(Type *...gameContext)
virtual bool alive()

Metodos públicos estaticos
static Moon::Alias::SystemType getSystemType ()

Miembros públicos
EntityStorage_t entities
ComponentStorage_t components

Descripcion detallada

Estabilidad

Estabilidad: 2 - Estable. La compatibilidad con todas las funciones de Estabilidad 2 y 3 es completa no se presentaran cambios en actulizaciones futuras

Ejemplo

```
struct PhysicsSys_t : Moon::Core::System_t<Gtx_t>
{
    inline static Moon::Tools::TimeStep_t time = Moon::Tools::TimeStep_t();
    void update(Gtx_t *gtx) override;

    bool alive() override;
};

//Use

//Create
auto physicsSys = PhysicsSys_t();

//Game Loop

while(...){
     .
     .
     .
     physicsSys.update(&gtx);
     renderSys.update(&gtx);
     inputSys.update();
     .
     .
     .
}

//Update Method
     void update(Gtx_t *gtx) override{
         for (Position2Df_t &cmp : gtx->getComponents<Position2Df_t>())
          {
              //Any Update cmp Physics
          }
     }
```

Contructores y Destructores

- **System_t**
  Constructor por defecto
  Definicion

```
template <MOON_IS_CTX_T... Type>
Moon::Core::System_t<Type>::System_t()
```

- **~System_t()**
  Destructor por defecto
  Definicion

```
template <MOON_IS_CTX_T... Type>
virtual Moon::Core::System_t< Type >::~System_t()
```

Funciones Miembro

- **void update(Type ...gameContext)* Actualiza el estado del GameContext_t recibido

```
/**
* @param gameContext
*/
template <MOON_IS_CTX_T... Type>
virtual void &Moon::Core::System_t::update(Type *...gameContext)
```

- **bool alive() = 0** Indica si el System_t esta activo

```
template <MOON_IS_CTX_T... Type>
bool alive() = 0
```