game Engine (ECS)

Engine

_systems : vector of Systems
_entities : vector of Entities
_firstUpdate : bool

update();

Entity& newEntity();
Entity& newEntity(entityType type);
Entity& addEntity(Entity &ent);
Entity& getEntity(int id);
bool killEntity(int id);
int genID();

System &getSystem(sysType type);
System &newSystem(sysType type);
System &getSystem3(sysType type);
bool hasSystem(sysType type);

Engine contains Entities and Systems

class Entity

_id : int
_parent : parent Engine *
_components : map of Components
_type : entityType

int getID();
Engine *getParent();
entityType getType();

bool hasComponent(compType type);
Component &getComponent(compType type);
void updateComponent(compType type, Component &comp);
void addComponent(compType type, Component &comp);
void removeComponent(compType type);

class Component

_type : compType

Each Entity contains a set of Components

System

_type : sysType
_parent : parent Engine *

Engine *getParent();
sysType getType();

virtual int run(Entity &ent);

R-type Implementation of Game Engine

Fight : public System

_type : sysType = ecs::FIGHT
_parent : parent Engine *

Engine *getParent();
sysType getType();

int run(Entity &ent);
void createMissile(rtype::Position &playerPos);

Move : public System

_type : sysType = ecs::MOVE
_parent : parent Engine *

Engine *getParent();
sysType getType();

int run(Entity &ent);
int run(Entity &ent, rtype::Position &add);
int checkForCollisions(Entity &entity);
int Collide(Entity &ent, Entity &hit);
void hitEntity(Entity &hit);
bool isDead(Entity &ent);
void addScore(int val)

Position : public Component

_type : compType = POSITION
_pos : rtype::vector2
_angle : int
_speed : float
_auto : bool

rtype::vector2 &getPos();
int getX();
int getY();
int getAngle();
float getSpeed();
bool isAutonomous();
void setPos(int x, int y);
void setPos(rtype::vector2 &pos);
void setAngle(int angle);
void setSpeed(float speed);
void setAutonomous(bool value);
void setAutonomous();

Position &operator+=(Position &other);
Position &operator-=(Position &other);
Position &operator=(Position &other);
bool operator==(Position &compare);
bool operator!=(Position &compare);

Graphical : Public Component

_type : compType = GRAPHICAL
_pos : rtype::vector2
_sprite : spriteType

rtype::vector2 &getSpritePos();
spriteType getSpriteType();
void setSpritePos(int x, int y);
void setSpritePos(rtype::vector2 &pos);
void setSpriteType(spriteType);

Graphical operator=(Graphical &compare);

Score : Public Component

_type : compType = SCORE
_score : int

void setSpritePos(int x, int y);
void setSpritePos(rtype::vector2 &pos);

Score &operator+=(Score &add);
Score &operator-=(Score &add);
Score &operator+=(int add);
Score &operator-=(int add);

Combat : public Component

_type : compType = COMBAT
_hp : unsigned int
_damage : usigned int
_missiles : unsigned int

unsigned int getHp();
unsigned int getDamage();
unsigned int getMissiles();
void setHp(unsigned int);
void setMissiles(unsigned int);
void setDamage(unsigned int);

void takeAHit();
bool fire();
bool fireSuper();

Combat &operator=(Combat &other);