ENTITY MANAGER

Functions

j1EntityManager()
virtual ~j1EntityManager()
bool Awake(xml_node&)
bool Start()
bool PreUpdate()
bool Update(dt)
void UpdateEntity(dt)
bool PostUpdate(dt)
bool CleanUp()
j1Entity * const EntityCreation(const char*, entity_type)
void EntityUpdate(float dt);
void KillEntity(j1Entity*)
void OnCollision(Collider1*, Collider2*)
Animation* LoadAnimation(const char*, const char*)
Playerdata& GetPlayerData() { return playerinfo }
SnakeData& GetSnakeData() { return snakeinfo }
BatData& GetBatData() { return batinfo }
bool Load(pugi::xml_node&)
bool Save(pugi::xml_node&) const

Variables

p2List <j1Entity*> entities
bool do_logic
int logic_updates_per_second
float update_ms_cycle
float accumulated_time
int entityID
bool loading
Playerdata playerinfo
SnakeData snakeinfo
BatData batinfo

*

ENTITY

enum entity_state
enum class entity_type

Functions

class j1Entity
j1Entity(const char*,entitytype) : manager, entitytype
void Init(j1EntityManager* manager)
virtual bool Start()
virtual bool Load(xml_node&)
virtual bool Save(xml_node&) const
virtual void FixedUpdate(float dt)
virtual void LogicUpdate(dt)
virtual bool CleanUp()
virtual void OnCollision(Collider1* , Collider2*)

Variables

p2SString name
fPoint Position
fPoint Velocity
fPoint Current_Velocity
Collider* Entity_Collider
SDL_Rect Entity_Collider_Rect
float colliding_offset
float Gravity
entity_type Entity_Type
entity_state Entity_State
int ID
Animation* CurrentAnimation
SDL_Texture* spritesheet
j1EntityManager* manager

Every entity has it's struct with their own variables. Only the relevant variables are shown

1

0..2

0..1

PLAYER

enum MODE
enum STATE
enum DIRECTION
enum MOVEMENT
struct PlayerData

Functions

j1Player();
~j1Player();
bool Start();
bool Update(float dt);
bool PostUpdate(float dt);
bool CleanUp();
void FixedUpdate(float dt);
void LogicUpdate(float dt);
void OnCollision(Collider * entitycollider, Collider * to_check);
void UpRight_Collision(Collider * entitycollider, Collider * to_check);
void Up_Collision(Collider * entitycollider, Collider * to_check);
void UpLeft_Collision(Collider * entitycollider, Collider * to_check);
void DownRight_Collision(Collider * entitycollider, Collider * to_check);
void Down_Collision(Collider * entitycollider, Collider * to_check);
void DownLeft_Collision(Collider * entitycollider, Collider * to_check);
void Right_Collision(Collider * entitycollider, Collider * to_check);
void Left_Collision(Collider * entitycollider, Collider * to_check);
bool Load(pugi::xml_node&);
bool Save(pugi::xml_node&) const;
Animation* LoadAnimation(const char* animationPath, const char* animationName);
SDL_Rect LoadColliderRect(const char* colliderPath, const char* colliderName);
void HandleMode();
void AddGravity(float dt);
void HandleState(float dt);
void GodModeMovement(float dt);
void StandingModeMovement(float dt);
void CrouchingModeMovement(float dt);
void Jump(float dt);
void DoubleJump(float dt);
void CheckDeath();
void CheckWin();
void CheckMovement();
void HandleAnimations();
void UpdateColliderPos();

Variables

PlayerData playerinfo;
MODE playermode;
STATE playerstate;
DIRECTION playerdirection;
MOVEMENT playermovement;
bool DoubleJumpAvailable;
bool CollidingGround;
bool CollidingPlatform;
bool CollidingLeftWall;
bool CollidingRightWall;
bool CollidingCeiling;
SDL_Rect Intersection;
fPoint Future_Position;
fPoint Player_Initial_Position;
fPoint Player_Displacement ;
bool SavedCheckPoint ;
fPoint LastCheckpointPostion ;
int SavedCheckPointArea;

SNAKE

struct SnakeData

Functions

j1Snake();
~j1Snake();
bool Start();
bool Update(float dt);
bool PostUpdate(float dt);
bool CleanUp();
void FixedUpdate(float dt);
void LogicUpdate(float dt);
void OnCollision(Collider* c1, Collider* c2);
bool Load(pugi::xml_node &config);
bool Save(pugi::xml_node &config) const;

Variables

bool dead ;
bool going_right ;
bool going_left ;
bool must_fall ;
bool Snakecolliding ;
SnakeData snakeinfo;

fPoint Snake_Collider_Margin

BAT

struct BatData

Functions

j1Bat();
~j1Bat();
bool Start();
bool Update(float dt);
bool PostUpdate(float dt);
bool CleanUp();
void FixedUpdate(float dt);
void LogicUpdate(float dt);
void OnCollision(Collider* c1, Collider* c2);
bool Load(pugi::xml_node &config);
bool Save(pugi::xml_node &config) const;
public:

Variables

bool dead ;
bool going_right ;
bool going_left ;
bool going_up ;
bool going_down ;
bool batcolliding ;
BatData batinfo;
fPoint Bat_Collider_Margin;