ENTITY MANAGER

Functions
j1EntityManager()
virtual -j1EntityManager()
bool Awake(xml_node&)
bool Start()
bool PreUpdate()
bool Update(dt)
void Update(dt)
void Update(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 basinfo }
bool Load(pugi: xml_node&)
bool Save(pugi:xml_node&)
bool Save(pugi:xml_node&)
const

Variables
p2Ust p2Ust cplest
p2Ust cplest
pool do_logic
int logic_updates_per_second
float update_ms_cycle
float accumulated_time
int entityID
bool loading
Playerdata playerinfo
SnakeData snakeinfo
BatData batinfo



enum entity_state enum class entity_type

Functions

Class | Tenthy

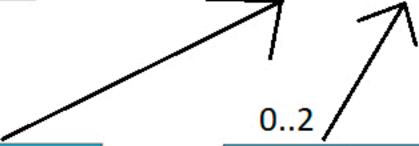
Class | Tenthy

(Class |

Variables

Variables
p255tring name
fPoint Position
fPoint Velocity
SPoint 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



PLAYER

enum MODE enum STATE

Functions

"j1Player();
bool Start();
bool Update(float dt);
bool PostUpdate(float dt);
bool CleanUp();
void FixedUpdate(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 Up_Collision(Collider * entitycollider, Collider * to_check);
void DownRight_Collision(Collider * entitycollider, Collider * to_check);
void DownRight_Collision(Collider * entitycollider, Collider * to_check);
void DownLeft_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);
void HandleMode();
void AddGravity(float dt);

void AddGravity(float dt); void HandleState(float dt); void GodModeMovement(float dt); void StandingModeMovement(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

Variables

PlayerData playerinfo;

MODE playermode;

STATE playerstate;

DIRECTION playerdirection;

MOVEMENT playermovement;

bool DoubleJumpAvailable;

bool CollidingPlatform;

bool CollidingPlatform;

bool CollidingRightWall;

bool CollidingRightWall;

bool CollidingCelling;

SDL_Rect Intersection;

(Point Player_Initial_Position;

(Point Player_Initial_Position;

(Point LastCheckpointPosition; Point LastCheckpointPostion int SavedCheckPointArea;

SNAKE

struct SnakeData

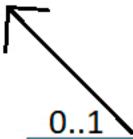
Functions

j1Snake(); ~j1Snake(); bool Start(): bool Update(float dt); bool Opdate(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(pugl::xml_node &config);
bool Save(pugl::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;