Inheritance

Association e.g : a use b: a use b

Aggregation

CGameObject

x, y: float
vx, vy: float
nx: int
state: int
isDeleted: bool
type: int

+SetPosition(float x, float y): void
+SetSpeed(float vx, float vy): void
+GetPosition(float &x, float &y): void
+GetSpeed(float &vx, float &vy): void
+GetState(): int

+Delete(): virtual void
+IsDeleted(): bool

+RenderBoundingBox(): void

CGameObject()
CGameObject(float x, float y, int type)

+GetBoundingBox(float &l, float &t, float &r, float &b): virtual void = 0
+Update(DWORD dt,
vector<LPGAMEOBJECT>* coObjects = NULL): virtual void
+Render(): virtual void = 0
+SetState(int state): virtual void

+IsCollidable(): virtual int
+OnNoCollision(DWORD dt): virtual void
+OnCollisionWith(LPCOLLISIONEVENT e): virtual void
+IsBlocking(): virtual int

~CGameObject()

+IsDeleted(const LPGAMEOBJECT &o): static bool
+GetTypeo()

use

public

CGoomba

ax, ay: float
die_start: ULONGLONG

+ GetBoundingBox(float &left, float &top, float &right, float &bottom): virtual void

Update(DWORD dt, vector<LPGAMEOBJECT> *coObjects): virtual void

Render(): virtual void

IsCollidable(): virtual int
IsBlocking(): virtual int
OnNoCollision(DWORD dt): virtual void

OnCollisionWith(LSCOLLISIONEVENT e)

+CGoomba(float x, float y, int type)
+SetState(int state): virtual void

public

CWingGoomba

-level: int
-walkingDistance: float
-startWalking/Location: float

+ GetBoundingBox(float &left, float &top, float &right, float &bottom): virtual void

Update(DWORD dt, vector<LPGAMEOBJECT> *coObjects): virtual void

Render(): virtual void

IsCollidable(): virtual int
IsBlocking(): virtual int
OnNoCollision(DWORD dt): virtual void

OnCollisionWith(LSCOLLISIONEVENT e)

+CWingGoomba(float x, float y, int type)
+SetState(int state): virtual void

+SetLevel(): void
+GetLevel(): int

public

CMushroom

ax, ay: float
id: int

GetBoundingBox(float &left, float &top, float &right, float &bottom): virtual void

Update(DWORD dt, vector<LPGAMEOBJECT> *coObjects): virtual void

Render(): virtual void

IsCollidable(): virtual int
IsBlocking(): virtual int
OnNoCollision(DWORD dt): virtual void

OnCollisionWith(LSCOLLISIONEVENT e)

+CMushroom(float x, float y, int type)
+SetState(int state): virtual void

public

CKoopa

ax, ay: float
shell_start: ULONGLONG

GetBoundingBox(float &left, float &top, float &right, float &bottom): virtual void

Update(DWORD dt, vector<LPGAMEOBJECT> *coObjects): virtual void

Render(): virtual void

IsCollidable(): virtual int
IsBlocking(): virtual int
OnNoCollision(DWORD dt): virtual void

OnCollisionWith(LSCOLLISIONEVENT e)

+CKoopa(float x, float y, int type)
+SetState(int state): virtual void

public

CInvisiblePlatform

-width, height: int

+CInvisiblePlatform(float x, float y, int type, int width, int height):CGameObject(x, y, type)

+Render(): void
+Update(DWORD dt): void
+GetBoundingBox(float l, float& t, float& r, float& b): void
+RenderBoundingBox(): void

public

CPlatform

length: int
cellWidth, cellHeight: float
spriteIdBegin, spriteIdMiddle, spriteIdEnd: int

+ CPlatform(float x, float y, int type, float cell_width, float cell_height, int length, int sprite_id_begin, int sprite_id_middle, int sprite_id_end):CGameObject(x, y, type)

+ Render(): void
+ Update(DWORD dt): void
+ GetBoundingBox(float& l, float& t, float& r, float& b): void
+ RenderBoundingBox(): void

public

CBrick

+ CBrick(float x, float y, int type): CGameObject(x, y, type)
+ Render(): void
+ Update(DWORD dt): void
+ GetBoundingBox(float& l, float& t, float& r, float& b): void
+ SetState(int state): void

public

CQuestionBrick

-by: float
-default_pos_y: float

-itemInsideId: int

+ CQuestionBrick(float x, float y, int type, int item_id): CBrick(x, y, type)
+ Render(): void
+ Update(DWORD dt, vector<LPGAMEOBJECT>* coObjects): void
+ GetBoundingBox(float& l, float& t, float& r, float& b): void
+ SetState(int state): void

public

CCoin

+ Coin(float x, float y, int type): CGameObject(x, y, type)
+ Render(): void
+ Update(DWORD dt): void
+ GetBoundingBox(float& l, float& t, float& r, float& b): void
+ IsBlocking(): int

public

CMario

-isSitting: BOOLEAN
-maxVx: float
-ax, ay: float // acceleration on x, y

-level: int
-untouchable: int
-untouchable_start: ULONGLONG
-isOnPlatform: BOOLEAN
-coin: int

isStopUpdate: bool

-OnCollisionWithGoomba(LPCOLLISIONEVENT e): void
-OnCollisionWithCoin(LPCOLLISIONEVENT e): void
-OnCollisionWithPortal(LPCOLLISIONEVENT e): void

-GetAniIdBig(): int
-GetAniIdSmall(): int

+CMario(float x, float y, int type): CGameObject(x, y, type)
+Update(DWORD dt, vector<LPGAMEOBJECT>* coObjects): void
+Render(): void
+SetState(int state): void

+IsCollidable(): int
+IsBlocking(): int

+OnNoCollision(DWORD dt): void
+OnCollisionWith(LPCOLLISIONEVENT e): void

+SetLevel(): void
+StartUntouchable(): void

+GetBoundingBox(float& left, float& top, float& right, float& bottom): void

+IsStopUpdate(): bool

public

CPortal

-scene_id: int // target scene to switch to
-width, height: float

+CPortal(float l, float t, float r, float b, int scene_id, int type)
+Render(): virtual void
+GetBoundingBox(float &l, float &t, float &r, float &b): virtual void
+GetScene(int): int
+IsBlocking(): ing

use

Utils

vector<string> split(string line, string delimeter = "|")

wstring ToWSTR(string st)

LPCWSTR ToLPCWSTR(string st)

Debug

void DebugOut(const wchar_t* fmt, ...)

void DebugOutTitle(const wchar_t* fmt, ...)

void SetDebugWindow(HWND hwnd)

AssetIDs

ID for textures

Object types

ID for sprites

use

CGame

-- _instance: static CGame*
-hWnd: Window handler

-backBufferWidth = 0: int
-backBufferHeight = 0: int

-pD3DDevice = NULL: ID3D10Device*
-pSwapChain = NULL: IDXGISwapChain*
-pRenderTargetView = NULL: ID3D10RenderView
-pBlendStateAlpha = NULL: ID3D10BlendState* // To store alpha blending state

-spriteObject: LPD3DX10SPRITE // Sprite handling object

-di: LPDIRECTINPUT8 // The Directinput object
-dikv: LPDIRECTINPUTDEVICE8 // The keyboard device

-keyState[KEYBOARD_STATE_SIZE]: BYTE // Directinput keyboard state buffer, size = 256
-keyEvents[KEYBOARD_BUFFER_SIZE]: DIDEVICEOBJECTDATA // Buffered keyboard data, size = 1024

-keyHandler: LPKEYHANDLEREVENT

-cam_x = 0.0f: float
-cam_y = 0.0f: float

-hInstance: HINSTANCE

-pPointSamplerState: ID3D10SamplerState*

-scenes: unordered_map<int, LPSCENE>
-current_scene: int
-next_scene = -1: int

+ _ParseSection_SETTINGS(string line): void
+ _ParseSection_SCENES(string line): void

+Init(HWND hWnd, HINSTANCE hInstance): void // Init DirectX, Sprite Handler

//
// Draw a portion or ALL the texture at position (x,y) on the screen. (x,y) is at the CENTER of the image
// rect : If NULL, the whole texture will be drawn
// If NOT NULL, only draw that portion of the texture
+Draw(float x, float y, LPTEXTURE tex, RECT* rect = NULL, float alpha = 1.0f)
+Draw(float x, float y, LPTEXTURE tex, int l, int t, int r, int b, float alpha = 1.0f)

+LoadTexture(LPCWSTR filePath): void

// Keyboard related functions
+InitKeyboard(): void
+IsKeyDown(int KeyCode): int
+ProcessKeyboard(): void
+SetKeyHandler(LPKEYEVENTHANDLER handler): void

+GetDirect3DDevice()
+GetSwapChain(): IDXGISwapChain*
+GetRenderTargetView(): ID3D10RenderTargetView*
+GetSpriteHandler(): ID3DX10Sprite*
+GetAlphaBlending(): ID3D10BlendState*
+int GetBackBufferWidth(): int
+int GetBackBufferHeight(): int

+GetInstance(): static CGame*

+SetPointSamplerState(): void

+SetCamPos(float x, float y): void
+GetCamPos(float& x, float& y): void

+Load(LPCWSTR gameFile): void
+GetCurrentScene() :LPSCENE
+SwitchScene(): void
+InitiateSwitchToScene(int scene_id): void

+ _ParseSection_TEXTURES(string line): void

~CGame()

use

CAnimations

-- _instance: static CAnimations*
-animations: unordered_map<int, LPANIMATION>

+Add(int id, LPANIMATION ani): void
+Get(int id): LPANIMATION
+Clear(): void

+GetInstance(): static CAnimation*

CAnimation

-lastFrameTime: ULONGLONG
-defaultTime: int
-currentFrame: int
-frames: vector<LPANIMATION_FRAME>

+CAnimation(int defaultTime = 100)
+Add(int spriteId, DWORD time = 0): void
+Render(float x, float y): void

CAnimationFrame

-sprite: LPSPRITE
-time: DWORD

+CAnimationFrame(LPSPRITE sprite, int time)
+GetTime(): DWORD
+GetSprite(): LPSPRITE

CTextures

-- _instance: static CTextures
textures: unordered_map<int, LPTEXTURE>

+CTextures()
+Add(int id, LPCWSTR filePath)
+Get(unsigned int i): LPTEXTURE

+GetInstance(): static CTextures*

CSprites

-- _instance: static CSprites*
-sprites: unordered_map<int, LPSPRITE>

+Add(int id, int left, int top, int right, int bottom, LPTEXTURE tex)
+Get(int id): LPSPRITE
+Clear(): void

+GetInstance(): static CPrites*

CTexture

_tex: ID3D10Texture2D*
rsview: ID3D10ShaderResourceView*
_width, _height: int

+CTexture()
+CTexture(ID3D10Texture2D* tex, ID3D10ShaderResourceView* rsview)

+getShaderResourceView(): ID3D10ShaderResourceView*
+getWidth(): int
+getHeight(): int

~CTexture()

link

CSprite

-id: int // sprite ID in sprite database
-left, top, right, bottom: int

-texture: LPTEXTURE
-sprite: D3DX10_SPRITE
-matScaling: D3DXMATRIX

+CSprite(int id, int left, int top, int right, int bottom, LPTEXTURE tex)
+GetLeft(): int
+GetTop(): int
+GetRight(): int
+GetBottom(): int
+Draw(float x, float y)

CCollision

+ -_instance: static CCollision*

+ SweptAABB(
float ml, mt, mr, mb,
float dx, dy,
float sl, st, sr, sb,
float &t, &nx, &ny): void

+ SweptAABB(
LPGAMEOBJECT objSrc,
DWORD dt,
LPGAMEOBJECT objDest): LPCOLLISIONEVENT

+ Scan(
LPGAMEOBJECT objSrc,
DWORD dt,
vector<LPGAMEOBJECT*> objDests,
vector<LPCOLLISIONEVENT>& coEvents): void

+ Filter(
LPGAMEOBJECT objSrc,
vector<LPCOLLISIONEVENT>& coEvents,
LPCOLLISIONEVENT &coX,
LPCOLLISIONEVENT &coY,
bool filterX,
int filterY,
int filterY): void

+ Process(
LPGAMEOBJECT objSrc,
DWORD dt,
vector<LPGAMEOBJECT*>* coObjects) void

+GetInstance(): static CCollision*

use

struct CCollisionEvent

src_obj: LPGAMEOBJECT // source object : the object from which to calculate collision
obj: LPGAMEOBJECT // the target object

t, nx, ny: float

dx, dy: float // *RELATIVE* movement distance between this object and obj
isDeleted: bool

CCollisionEvent(float t, float nx, float ny, float dx = 0, float dy = 0,
LPGAMEOBJECT obj = NULL, LPGAMEOBJECT src_obj = NULL)
WasCollided(); int
compare(const LPCOLLISIONEVENT& a, LPCOLLISIONEVENT& b)

have

CScene

key_handler:
LPKEYEVENTHANDLER
id: int
sceneFilePath: LPCWSTR

+CScene(int id, LPCWSTR filePath)

+GetKeyEventHandler():
LPKEYEVENTHANDLER
+Load(): virtual void = 0
+UnLoad(): virtual void = 0
+Update(DWORD dt): virtual void = 0
+Render(): virtual void = 0

have

CKeyEventHandler

+KeyState(BYTE* state): virtual void = 0
+OnKeyDown(int KeyCode): virtual void = 0
+OnKeyUp(int KeyCode): virtual void = 0

public

CSceneKeyHandler

scene: CScene*

CSceneKeyHandler(LPSCENE s): CKeyEventHandler()

+KeyState(BYTE* state): virtual void = 0
+OnKeyDown(int KeyCode): virtual void = 0
+OnKeyUp(int KeyCode): virtual void = 0

public

CPlayScene

player: LPGAMEOBJECT
objects: vector<LPGAMEOBJECT>
map/image: LPIMAGEMAP

_ParseSection_SPRITES(string line): void
_ParseSection_ANIMATIONS(string line): void
_ParseSection_ASSETS(string line): void
_ParseSection_OBJECTS(string line): void

LoadAssets(LPCWSTR assetFile): void

+CPlayScene(int id, filePath LPCWSTR)
+Load(): virtual void
+UnLoad(): virtual void
+Update(DWORD dt): void
+Render(): void

+GetPlayer(): LPGAMEOBJECT

+Clear(): void
+PurgeDeletedObjects(): void

+IsGameObjectDeleted(const LPGAMEOBJECT& o)

public

CSampleKeyHandler

CSampleKeyHandler(LPSCENE s): CSceneKeyHandler(s)

+KeyState(BYTE* state): virtual void
+OnKeyDown(int KeyCode): virtual void
+OnKeyUp(int KeyCode): virtual void

have

CTile

-x, y: int
-sprite_id: int

+CTile()
+CTile(int x, int y, int id)
+Render(): void

CTileLayer

-layer: vector<LPTILE>

+CTileLayer(vector<LPTILE> tiles)
+Render()

CTileMap

// layer's draw order, layer
-layers: unordered_map<int, LPTILELAYER>

+CTileMap(string filePath)
+Render()
+Clear()