Inheritance \_\_\_\_\_> AssetIDs Debug vector<string> split(string line, stringdelimeter = "\t") ID for textures void DebugOut(const wchar\_t\* fmt, ...) wstring ToWSTR(string st) Object types void DebugOutTitle(const wchar\_t\* fmt, ...) Aggregation ——— LPCWSTR ToLPCWSTR(string st) ID for sprites void SetDebugWindow(HWND hwnd) CCollision \_\_instance: static CCollision\* + SweptAABB( float ml, mt, mr, mb, float dx, dy, float sl, st, sr, sb, float &t, &nx, & ny): void **CGameObject** + SweptAABB( LPGAMEOBJECT objSrc, struct CCollisionEvent src obj: LPGAMEOBJECT // source object : the object from which to calculate collision LPGAMEOBJECT objDest): LPCOLLISIONEVENT obj: LPGAMEOBJECT // the target object # vx, vy: float # nx: int LPGAMEOBJECT objSrc, t, nx, ny: float # state: int DWORD dt, dx, dy: float // \*RELATIVE\* movement distance between this object and obj vector<LPGAMEOBJECT>\* objDests, # isDeleted: bool isDeleted: bool vector<LPCOLLISIONEVENT>& coEvents): void CCollisionEvent(float t, float nx, float ny, float dx = 0, float dy = 0, LBGAMEOBJECT objSrc, CGame LPGAMEOBJECT obj = NULL, LPGAMEOBJECT src\_obj = NULL) vector<LPCOLLISIONEVENT>& coEvents, +SetPosition(float x, float y): void WasCollided(): int LPCOLLISIONEVENT &colX, -\_\_instance: static CGame\* compare(const LPCOLLISIONEVENT& a, LPCOLLISIONEVENT& b) +SetSpeed(float vx, float vy): void LPCOLLISIONEVENT &coly, -hWnd: Window handler +GetPosition(float &x, float &y): void int filterBlock, -backBufferWidth = 0: int int filterX, +GetSpeed(float &vx, float &vy): void -backBufferHeight = 0: int int filterY): void +GetState(): int -pD3DDevice = NULL: ID3D10Device\* + Process( LPGAMEOBJECT objSrc, +Delete(): virtual void -pSwapChain = NULL: IDXGISwapChain\* -pRenderTargetView = NULL: ID3D10TargetView +IsDeleted(): bool vector<LPGAMEOBJECT>\* coObjects) void -pBlendStateAlpha = NULL: ID3D10BlendState\* // To store alpha blending state +GetInstance(): static CCollision\* +RenderBoundingBox() -spriteObject: LPD3DX10SPRITE // Sprite handling object -di: LPDIRECTINPUT8 // The DirectInput object CGameObject() -didv: LPDIRECTINPUTDEVICE8 // The keyboard device CGameObject(float x, float y) -keyStates[KEYBOARD\_STATE\_SIZE]: BYTE // DirectInput keyboard state buffer, size = 256 -keyEvents[KEYBOARD\_BUFFER\_SIZE]: DIDEVICEOBJECTDATA // Buffered keyboard data, size = 1024 +GetBoundingBox(float &I;, float &t, float &r, float &b): virtual void = 0 -keyHandler: LPKEYHANDLEREVENT vector<LPGAMEOBJECT>\* coObjects = NULL): virtual void  $-cam_x = 0.0f$ : float +Render(): virtual void = 0  $-cam_y = 0.0f$ : float +SetState(int state): virtual void -hInstance: HINSTANCE +IsCollidable(): virtual int -pPointSamplerState: ID3D10SamplerState\* +OnNoCollision(DWORD dt): virtual void +OnCollisionWith(LPCOLLISIONEVENT e): virtual void +IsBlocking(): virtual int -current\_scene: int CScene -next\_scene = -1: int ~CGameObject() # key\_handler: -\_ParseSection\_SETTINGS(string line): void LPKEYEVENTHANDLER -\_ParseSection\_SCENES(string line): void # id: int CKeyEventHandler +IsDeleted(const LPGAMEOBJECT &o): static bool # sceneFilePath: LPCWSTR +Init(HWND hWnd, HINSTANCE hInstance): void // Init DirectX, Sprite Handler +CScene(int id, LPCWSTR filePath) –have— // Draw a portion or ALL the texture at position (x,y) on the screen. (x,y) is at the CENTER of the image +KeyState(BYTE\* state): virtual void = 0 +GetKeyEventHandler(): // rect : if NULL, the whole texture will be drawn +OnKeyDown(int KeyCode): virtual void = 0 LPKEYEVENTHANDLËR / if NOT NULL, only draw that portion of the texture +OnKeyUp(int KeyCode): virtual void = 0 +Load(): virtual void = 0 +Draw(float x, float y, LPTEXTURE tex, RECT\* rect = NULL, float alpha = 1.0f) +UnLoad(): virtual void = 0 +Draw(float x, float y, LPTEXTURE tex, int I, int t, int r, int b, float alpha = 1.0f) +Update(DWORD dt) virutal void = 0 +Render(): virutal void = 0 +LoadTexture(LPCWSTR filePath): void // Keyboard related functions +IniKeyboard(): void +IsKeyDown(int KeyCode): int CSceneKeyHandler +ProcessKeyboard(): void +SetKeyHandler(LPKEYEVENTHANDLER handler): void # scene: CScene\* +GetDirect3DDevice() +GetSwapChain(): IDXGISwapChain\* CSceneKeyHandler(LPSCENE s) :CKeyEventHandler() +GetRenderTargetView(): ID3D10RenderTargetView\* +GetSpriteHandler(): ID3DX10Sprite\* +KeyState(BYTE\* state): virtual void = 0 +GetAlphaBlending(): ID3D10BlendState\* +OnKeyDown(int KeyCode): virtual void = 0 CPlayScene +int GetBackBufferWidth(): int +OnKeyUp(int KeyCode): virtual void = 0 +int GetBackBufferHeight(): int # player: LPGAMEOBJECT # objects: vector<LPGAMEOBJECT> +GetInstance(): static CGame\* # mapImage: LPIMAGEMAP +SetPointSamplerState(): void # ParseSection\_SPRITES(string line): void **CPlatform** CGoomba ClmageMap CCoin # \_ParseSection\_ANIMATIONS(string line): void +SetCamPos(float x, float y): void # ax, ay: float -isSitting: BOOLEAN -scene\_id: int // target scene to switch to -spriteId: int # \_ParseSection\_ASSETS(string line): void +GetCamPos(float& x, float & y): void CSampleKeyHandler # die\_start: ULONGLONG # cellWidth, cellHeight: float -maxVx: float -width, height: float # \_ParseSection\_OBJECTS(string line): void # spriteIdBegin, spriteIdMiddle, spriteIdEnd: int -ax, ay: float // accelaration on x, y +Load(LPCWSTR gameFile): void +CImageMap(float x, float, y, int id):CGameObject(x, y) CBrick(float x, float y): CGameObject(x, y) + Ccoin(float x, float y): CGameObject(x, y) # LoadAssets(LPCWSTR assetFile): void +GetCurrentScene(): LPSCENE # GetBoundingBox(float &left, float &top, float &right, float Render(): void + Render(): void +CPortal(float I, float t, float r, float b, int -level: int +SwithScene(): void + Update(DWORD dt): void +Update(DWORD dt): void &bottom): virtual void + CPlatform(float x, float y, float cell\_width, float cell\_height, int Update(DWORD dt): void scene\_id) CSampleKeyHandler(LPSCENE s) :CSceneKeyHandler(s) -untouchable: int +CPlayScene(int id, filePath LPCWSTR) +InitiateSwitchScene(int scene\_id): void +Render(): void length, int sprite\_id\_begin, int sprite\_id\_middle, int sprite\_id\_end) GetBoundingBox(float& I, float& t, float& r, float& b): + GetBoundingBox(float& I, float& t, float& r, float& b): +Render(): virtual void -untouchable\_start: ULONGLONG +Load(): virtual void + GetBoundingBox(float& I, float& t, float& r, float& b): void # Update(DWORD dt, vector<LPGAMEOBJECT> :CGameObject(x, y) +GetBoundingBox(float &I, float &t, float &r, +KeyState(BYTE\* state): virtual void -isOnPlatform: BOOLEAN +UnLoad(): virutal void +\_ParseSection\_TEXTURES(string line): void \*coObjects): virtual void + IsBlocking(): int float &b): virtual void +OnKeyDown(int KeyCode): virtual void +Update(DWORD dt): void -coin: int +GetSceneId(): int + Render(): void +OnKeyUp(int KeyCode): virtual void ~CGame() +Render(): void # Render(): virtual void + Update(DWORD dt): void +IsBlocking(): ing + GetBoundingBox(float& I, float& t, float& r, float& b): void -OnCollisionWithGoomba(LPCOLLISIONEVENT e): void +GetPlayer(): LPGAMEOBJECT # IcCollidable(): virtual int -OnCollisionWithCoin(LPCOLLISIONEVENT e): void # IsBlocking():: virtual int -OnCollisionWithPortal(LPCOLLISIONEVENT e): void +Clear(): void # OnNoCollision(DWORD dt): virtual void +PurgeDeletedObjects(): void -GetAniIdBig(): int # OnCollisionWith(LSCOLLISIONEVENT e) -GetAniIdSmall(): int +IsGameObjectDeleted(const LPGAMEOBJECT& o) +CMario(float x, float y) : CGameObject(x, y) +CGoomba(float x, float y) +SetState(int state): virtual void +Update(DWORD dt, vector<LPGAMEOBJECT>\* coObjects): void() +Render(): void +SetState(int state): void +IsCollidable(): int +IsBlocking(): int **CAnimations** +OnNoCollision(DWORD dt): void -\_\_instance: static CAnimations\* +OnCollisionWith(LPCOLLISIONEVENT e): void -animations: unordered\_map<int,</pre> LPANIMATION> +SetLevel(int I): void +StartUntouchable(): void +Add(int id, LPANIMATION ani): void +GetBoundingBox(float& left, float& top, float& right, +Get(int id): LPANIMATION float& bottom): void +Clear(): void +GetInstance(): static CAnimation\* // layer's draw order, layer CTileLayer -layers: unordered\_map<int, LPTILELAYER> -x, y: int -layer: vector<LPTILE> -sprite\_id: int CTileMap(string filePath) +CTile() +CTileLayer(vector<LPTILE> tiles) +Render() +CTile(int x, int y, int id) +Render() +Clear() **CAnimation** +Render(): void -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 CTextures -\_\_instance: static CTextures CSprites textures: unordered\_map<int, CAnimationFrame LPTEXTURE> \_instance: static CSprites\* -sprite: LPSPRITE -sprites: unordered\_map<int, LPSPRITE> -time: DWORD +CTextures() +Add(int id, LPCWSTR filePath) +Add(int id, int left, int top, int right, int bottom, +Get(unsigned int i): +CAnimationFrame(LPSPRITE LPTEXTURE tex) LPTEXTURE sprite, int time) +Get(int id): LPSPRITE +GetTime(): DWORD +Clear(): void +GetInstance(): static CTextures\* +GetSprite(): LPSPRITE +GetInstance(): static CPrites\* CTexture CSprite # \_tex: ID3D10Texture2D\* # rsview: ID3D10ShaderResourceView\* -id: int // sprite ID in sprite database # \_width, \_height: int -left, top, right, bottom: int +CTexture() -texture: LPTEXTURE +CTexture(ID3D10Texture2D\* tex, -sprite: D3DX10\_SPRITE ID3D10ShaderResourceView\* rsview) -matScaling: D#DXMATRIX +getShaderResourceView(): ID3D10ShaderResourceView\* +CSprite(int id, int left, int top, int right, +getWidth(): int int bottom, LPTEXTURE tex) +getHeight(): int +Draw(float x, float y) ~CTexture()