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 &I;, 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 +GetType() **CPlatform CPortal** CGoomba **CBrick** # ax, ay: float -isSitting: BOOLEAN -width, height: int # length: int -scene_id: int // target scene to switch to # cellWidth, cellHeight: float -width, height: float # die_start: ULONGLONG -maxVx: float # spriteIdBegin, spriteIdMiddle, spriteIdEnd: int -ax, ay: float // accelaration on x, y +CInvisiblePlatform(int x, int y, int type, int width, CBrick(float x, float y, int type): CGameObject(x, y, + Ccoin(float x, float y, int type): CGameObject(x, y, # GetBoundingBox(float &left, float &top, float &right, float int height):CGameObject(x, y, type) +CPortal(float I, float t, float r, float b, int -level: int &bottom): virtual void + CPlatform(float x, float y, int type, float cell_width, float cell_height, - Render(): void + Render(): void scene_id, int type) -untouchable: int +Render(): void int length, int sprite_id_begin, int sprite_id_middle, int sprite_id_end) - Update(DWORD dt): void + Update(DWORD dt): void +Render(): virtual void -untouchable_start: ULONGLONG :CGameObject(x, y, type) # Update(DWORD dt, vector<LPGAMEOBJECT> +Update(WORD dt): void + GetBoundingBox(float& I, float& t, float& r, float& b): + GetBoundingBox(float& I, float& t, float& r, float& b): +GetBoundingBox(float &I, float &t, float &r, -isOnPlatform: BOOLEAN +GetBoundingBox(float& I, float& t, float& r, float& *coObjects): virtual void float &b): virtual void -coin: int + Render(): void + IsBlocking(): int +GetSceneId(): int + Update(DWORD dt): void # Render(): virtual void +RenderBoundingBox(): void +IsBlocking(): ing + GetBoundingBox(float& I, float& t, float& r, float& b): void -OnCollisionWithGoomba(LPCOLLISIONEVENT e): void # IcCollidable(): virtual int + RenderBoundingBox(): void -OnCollisionWithCoin(LPCOLLISIONEVENT e): void -OnCollisionWithPortal(LPCOLLISIONEVENT e): void # IsBlocking():: virtual int # OnNoCollision(DWORD dt): virtual void -GetAniIdBig(): int # OnCollisionWith(LSCOLLISIONEVENT e) -GetAniIdSmall(): int CQuestionBrick +CGoomba(float x, float y, int type) +CMario(float x, float y, int type) : CGameObject(x, y, +SetState(int state): virtual void +Update(DWORD dt, vector<LPGAMEOBJECT>* coObjects): void() + CQuestionBrick(float x, float y, int type): CBrick(x, y, type) +Render(): void + Render(): void +SetState(int state): void + Update(DWORD dt): void + GetBoundingBox(float& I, float& t, float& r, float& b): void +IsCollidable(): int +IsBlocking(): int +OnNoCollision(DWORD dt): void +OnCollisionWith(LPCOLLISIONEVENT e): void +SetLevel(int I): void +StartUntouchable(): void +GetBoundingBox(float& left, float& top, float& right, float& bottom): void

