**GAME**

1

**Entity Manager**
------------------------------------------------------------

Entity* entity_array[MAX_ENTITYES];
Enemy_pointers (*Assumption 1*)
Partticle_pointers (*Assumption 2*)

------------------------------------------------------------

j1EntityManager()
~j1EntityManager()

Entity* CreateEntity(Entity::EntityTypes type, int x, int y)
void DestroyDeletedEntity()
void Destroy_all()
bool Save(pugi::xml_node& data) const
bool Load(pugi::xml_node& data)

bool Awake(pugi::xml_node&)
bool Start()
bool PreUpdate()
bool Update(float dt)
bool PostUpdate()
bool CleanUp()

void OnCollision(Collider* c1, Collider* c2); (*Assumption 3*)

*

**Player**
--------------------------------------------------------------------------------------

*Assumption 3,5, 6, 7, 8*
PLAYER_STATES state
p2Qeue<PLAYER_INPUTS> inputs
Collider* collider
iPoint position_buffer
Animation* current_animation
--------------------------------------------------------------------------------------

Player(int x, int y)
~Player()

bool Start()
bool Update(float dt)

void Draw(float dt)

void internal_input(p2Qeue<PLAYER_INPUTS>& inputs, float dt)
bool external_input (p2Qeue<PLAYER_INPUTS> & inputs, float dt)
PLAYER_STATES process_fsm (p2Qeue<PLAYER_INPUTS>& inputs)
void OnCollision(Collider* c2)

void playerJump(PLAYER_STATES state, float dt)
void playerFall(float dt)
bool ResetJumpCheck(SDL_Rect player, SDL_Rect collision)
void Check_if_falling()
void PlayerMov(float dt, float factor = 1)
void Stay_in_map(SDL_Rect rect)

1

**Entity**
-----------------------------------------------------------

*Assumption 4*
iPoint position
EntityTypes type
SDL_Texture* texture
bool to_delete
Animation animation
bool started
-----------------------------------------------------------

virtual bool Update(float dt)
virtual bool Start()

virtual bool HandleInput()

virtual void Draw(float dt)

virtual void OnCollision(Collider* collider)
virtual Collider* GetCollider() const

0..

**Particle**
-------------------------
float life
-------------------------
~Particle()
void Draw(float dt)

0..1

**Explosion**
-------------------------
Collider* collider
-------------------------
Explosion(int x, int y)
bool Start()
bool Update(float dt)

0..

0..

**Enemy**
-----------------------------------------------------------
Collider* collider
iPoint position_buffer
bool in_path
float frames_stuck
p2DynArray<iPoint>* path
float path_counter
*Assumption 5*
-----------------------------------------------------------
~Enemy()
void Destroy()

virtual void Move(iPoint pos, float dt)
virtual void LockOn(iPoint destiny, float dt)
virtual bool CheckLockOn(iPoint destiny)
virtual void AvoidStuck(iPoint destiny)
virtual void CheckStuck()
virtual void Pathfind(float dt)
virtual void FollowPath(float dt)
virtual void ResetPathCounter(float dt)

bool Update(float dt);
bool HandleInput();

void Draw(float dt);

0..

**Walking_Enemy**
-------------------------------------------
bool going_up
float speed
-------------------------------------------
Walking_Enemy(int x, int y)

void OnCollision(Collider* collider)
bool Start()

void Move(iPoint destiny, float dt)
void LockOn(iPoint destiny, float dt)
void FollowPath(float dt)
void AvoidStuck(iPoint destiny)
void CheckStuck()
bool CheckLockOn(iPoint destiny)

**Dust**
-------------------------
-------------------------
Dust(int x, int y)
bool Start()
bool Update(float dt)

0..

**Alien_Enemy**
-------------------------------------------
float wave
bool going_up
-------------------------------------------
Alien_Enemy(int x, int y)

void Move(iPoint destiny, float dt)
void LockOn(iPoint destiny, float dt)
void OnCollision(Collider* collider)
bool CheckLockOn(iPoint destiny)
bool Start()

**Assumptions**

General: All entities with a collider know the
function Collider* GetCollider() const

1: We have all pointers necessary to all enemy related
values

2:We have all pointers necessary to all particle related
values

3: We have all functions necessary to perform OnCollision
(such as check direction, etc.)

4: We have an enum with all entity types

5: Presence of non-specific logic-related bools, such as
bool   flip to know if we need to draw the texture flipped or
not

6: We have the enums of Explosion_Direction,
Player_States and Player_Inputs

7: We have all variables "int" or "float" necessary to perform
player functions (such as speed, time spent jumping, CD
from last explosion, etc.)

8: We do have pointers to all animations and textures used
in player (walking animation, jumping animation, etc.)