GameObject

Code

```
enum                                                                    GameObjectType
{
 TYPE_UNKNOWN,
 TYPE_OBJECT,
 TYPE_PERSON,
 TYPE_HOUSE,
 TYPE_VEHICLE,
 TYPE_OPEN_HOUSE
};
```

Code

```
enum                                                                    FloorPlacement
{
 PLACEMENT_NONE,    //        frei        dreh-        und        positionierbar
 PLACEMENT_AXISALIGNED,  //Objektwirdnichtgedreht,Pivotpunktmit0Offsetentscheidetdie0Höhe
 PLACEMENT_BOXALIGNED,   //Objektfreidrehbar,Boundingbox-Auflagepunktmit0Offsetentscheidetüber0Höhe
 PLACEMENT_ALIGNED_CORNERS, //Eckeder Boundingboxentscheidetüber0Höhe0Ausrichtung,untersteEckezählt
 PLACEMENT_ALIGNED_SAMPLED, //   genauer   als   boundingcorners,   zeitaufwendiger
 PLACEMENT_PHYSICS,   //    physik    benutzen.    Nicht    im    Game    möglich!
 PLACEMENT_CUSTOM_PLACEMENT //placementwirdvonabgeleiteterKlasseberechnet(e.g.Person,Vehicle)
};
```

Code

```
enum                                                            PhysicCollisionMode
{
 PHYSIC_COLLISION_ALL                              =                               0,
 PHYSIC_COLLISION_NOWORLD                            =                             1,
 PHYSIC_COLLISION_NOEXTERNALBODIES                        =                          2,
 PHYSIC_COLLISION_NOINTERNALBODIES                        =                          4,
 PHYSIC_COLLISION_USECATEGORIES                         =                            8,
 PHYSIC_COLLISION_NOTRACELINE                            =                        16,
 PHYSIC_COLLISION_NORESPONSE                             =                        32,
 PHYSIC_COLLISION_NONE  =  ~PHYSIC_COLLISION_ALL  &  ~PHYSIC_COLLISION_USECATEGORIES
};
```

Alles anzeigen
Code

```
enum                                                                    EnergyType
{
 ENERGYTYPE_UNKNOWN,
 ENERGYTYPE_FIRE,
 ENERGYTYPE_SHOT,
 ENERGYTYPE_WATER,
 ENERGYTYPE_POLICEBLOCK,
 ENERGYTYPE_CARPUSH,
 ENERGYTYPE_PHYSIC,
 ENERGYTYPE_WATER_FROM_OUSIDE,
 ENERGYTYPE_WATER_FROM_INSIDE,
 ENERGYTYPE_SHOTSOUND,
 ENERGYTYPE_FLIGHTSHOT
};
```

Alles anzeigen
Code

```
//                  constants                   for                   object-flags
enum                                                                    ObjectFlag
{
 OF_NONE      =                                0x00000000,
 OF_PERSON_ENCLOSED   =    0x00000001, //    object    has    enclosed    person
 OF_LOCKED     =       0x00000002, //       object       is       locked
 OF_USABLE     =   0x00000004, //   object   can   be   used   (special   action)
 OF_BULLDOZABLE    = 0x00000008, // can be planated by the bulldozer (wheel loader)
 OF_TRANSPORTABLE   =  0x00000010, //  object  can  be  loaded  up  (by  vehicle)
 OF_PULLABLE     =      0x00000020, //      object      can      be      pulled
 OF_ACCESSIBLE    =     0x00000040, //     object     can     be     entered
 OF_COOLABLE     =  0x00000080,       //  object  can  be  cooled  by  fire  fighters
 OF_SHOOTABLE    =    0x00000100, //    object    can    be    shooted    at
 OF_CUTABLE     =   0x00000200, //   object   can   be   cut   down   with   chainsaw
 OF_USABLE_WITH_MEGAPHONE =  0x00000400, //  megaphone  can  be  used  on  object
 OF_RECOVERABLE    =  0x00000800, //  object  can  be  recovered  by  recovery  crane
 OF_FLOTSAM     =   0x00001000, //   object   can   be   picked   up   by   motor   boat
 OF_HIDDEN     =       0x00002000, //       object       is       invisible
 OF_CARRYABLE_BY_BULLDOZER =0x00008000, //canbecarriedbythebulldozer(wheelloader)
 OF_HAS_FIRE_EXTINGUISHER = 0x00010000, // object carries one/many fire extinguishers
 OF_HAS_SHEARS    =   0x00020000, //   object   carries   one/many   jaws   of   life
 OF_HAS_CHAINSAW    =    0x00040000, //    object    carries    one/many    chainsaws
 OF_HAS_HOSE     =    0x00080000, //    object    carries    one/many    fire    hoses
 OF_HAS_JUMPPAD    =    0x00100000, //    object    carries    one/many    jumppads
 OF_HAS_ROADBLOCK   =    0x00200000, //   object   carries   one/many   roadblocks
 OF_HAS_FLASHGRENADE   =  0x00400000, //  object  carries  one/many  flashgrenades
 OF_HAS_FIREAXE    =    0x00800000, //    object    carries    one/many    fireaxes

 OF_BLOCKED     =    0x80000000 //    no    interaction    with    object    possible
};
```

Alles anzeigen

Code

```
enum                                                            ObjectBurnStatus
{
 OBS_NORMAL   =  0x01, //  object  is  in  normal  mode  /  visible  in  normal  mode
 OBS_HALFBURNED  =  0x02, //  object  is  halfburned  /  visible  in  halfburned  mode
 OBS_FULLBURNED  =  0x04 //  object  is  fullburned  /  visible  in  fullburned  mode
};
```

Code

```
enum                                                            TrafficLightType
{
 TLT_NONE,
 TLT_GREEN,
 TLT_YELLOW,
 TLT_RED
};
```

Code

```
enum                                                            BlinkerLightType
{
 BLT_NONE,
 BLT_LEFT,
 BLT_RIGHT,
 BLT_BOTH
};
```

Code

```
enum HaltType
{
 HALT_PERSONS,
 HALT_VEHICLES,
 HALT_BOTH
};
```

Code

```
enum                                                    ContaminationType
{
 CONTAMINATION_ATOMIC,
 CONTAMINATION_CHEMICAL,
 CONTAMINATION_BIOLOGICAL
};
```

Code

```
union                                                    ActionParameters
{
 float                                                             fValue;
 int                                                               iValue;
 bool                                                              bValue;
 char                                                             *cValue;
 class                                  Actor                     *aValue;
 void                                                             *pValue;
};
```

Code

```
struct                                                     ActionCallback
{
 class                                GameObject                    *Owner;
 ActionParameters                                            Parameters[8];
 ActionCallback();
};
```

Code

```
class GameObject : public Actor
{
 GameObject();
 GameObject(const GameObject &Obj_);
 GameObject(const Actor *Act_);
 virtual ~GameObject();
 virtual bool IsValid() const;

 bool HasAnimation(const char *Anim_);
 void SetAnimation(const char *Anim_);
 bool IsCurrentAnimation(const char *Anim_);
 bool IsAnimationFinished() const;

 bool AssignCommand(const char *Command_);
 void RemoveCommand(const char *Command_);
 void EnableCommand(const char *Command_, bool Enabled_);
 void DisableAllCommands();
 bool IsCommandEnabled(const char *Command_);
 bool HasCommand(const char *Command_);
 void SetCommandable(bool Commandable_);

 void EnableOnContactCallback(bool enable_);
 GameObjectType GetObjectType() const;
 GameObjectList GetCarriedObjects();

 virtual void UpdatePlacement(void);
 virtual void SetPlacementNone(void);
 virtual void SetPlacement(FloorPlacement placement_);
 virtual void ComputePlacement(float &x_, float &y_, float &z_);

 virtual void SetPosition(const Vector &v_);
 virtual void SetPosition(const GameObject* obj_);
 virtual void SetRotation(const GameObject* obj_);
 virtual void SetRotation(float r00, float r01, float r02,
                          float r10, float r11, float
                          float r20, float r21, float
 virtual void SetRotationPutInXY(float r00, float r01, float r02,
        float r10, float r11, float r12,
        float r20, float r21, float r22);
 virtual void GetRotation(float &r00, float &r01, float &r02,
                          float &r10, float &r11, float
                          float &r20, float &r21, float

 virtual Vector GetPosition() const;

 virtual Vector GetLookatDir();
 virtual void GetLookatDir(float &x_, float &y_, float &z_);
 virtual void SetLookatDir(float x_, float y_, float z_);
 virtual void SetLookatDir(Vector &dir_);

 virtual void GetModelPosition(float &x_, float &y_, float &z_);
 virtual float GetPositionOffset();
 const char *GetPrototypeName();
 const char *GetPrototypeFileName();
 const char *GetModelFileName();

 bool IsInsideMap();
 void GetOrientedBBoxPoint( unsigned int corner_, float &x_, float &y_, float &z_);
 void DrawOrientedBBox(unsigned char red_=255, unsigned char green_=255, unsigned char blue_=255);

 bool IsJumppad() const;
 bool IsRoadblock() const;
 bool IsHydrant() const;
 bool IsHydrantInUse() const;
 bool IsInsideWater() const;
 bool IsIdle() const;
 bool CanStopAction()
 bool IsWaiting() const;
 bool HasFireChilds() const;
 bool IsSelected() const;
 bool IsShooted() const;
```

Alles anzeigen