SOBJECTSTATE Structure

C++

```
struct SOBJECTSTATE {
  enum EFireControllerIndex {
    eFireControllerIndex_None = 0,
    eFireControllerIndex_First = 1,
    eFireControllerIndex_Second = 2,
    eFireControllerIndex_Third = 3,
    eFireControllerIndex_Fourth = 4,
    eFireControllerIndex_All = 17185
  };
  bool jump;
  bool aimObstructed;
  bool aimTargetIsValid;
  bool forceWeaponAlertness;
  EAIFireState fire;
  EAIFireState fireSecondary;
  EAIFireState fireMelee;
  ERequestedGrenadeType requestedGrenadeType;
  int weaponAccessories;
  int bodystate;
  float lean;
  float peekOver;
  Vec3 vShootTargetPos;
  Vec3 vAimTargetPos;
  Vec3 vLookTargetPos;
  Vec3 vLobLaunchPosition;
  Vec3 vLobLaunchVelocity;
  SFireCommandProjectileInfo projectileInfo;
  ELookStyle eLookStyle;
  bool bAllowLowerBodyToTurn;
  Vec3 vMoveDir;
  Vec3 vForcedNavigation;
  Vec3 vBodyTargetDir;
  Vec3 vDesiredBodyDirectionAtTarget;
  float fForcedNavigationSpeed;
  unsigned int movementContext;
  SAIPredictedCharacterStates predictedCharacterStates;
  Vec3 vMoveTarget;
  Vec3 vInflectionPoint;
  float fMovementUrgency;
  float fDesiredSpeed;
  float fTargetSpeed;
  bool allowStrafing;
  bool allowEntityClampingByAnimation;
  float fDistanceToPathEnd;
  Vec3 vDirOffPath;
  EActorTargetPhase curActorTargetPhase;
  Vec3 curActorTargetFinishPos;
  SAIActorTargetRequest actorTargetReq;
  bool bCloseContact;
  bool bReevaluate;
  bool bTargetEnabled;
  bool bTargetIsGroupTarget;
  bool continuousMotion;
  EAITargetType eTargetType;
  EAITargetContextType eTargetContextType;
  EAITargetThreat eTargetThreat;
  tAIObjectID eTargetID;
  EAITargetType ePeakTargetType;
  EAITargetThreat ePeakTargetThreat;
  tAIObjectID ePeakTargetID;
  EAITargetType ePreviousPeakTargetType;
  EAITargetThreat ePreviousPeakTargetThreat;
  tAIObjectID ePreviousPeakTargetID;
```

```
  Vec3 vTargetPos;
  float fDistanceFromTarget;
  EAITargetStuntReaction eTargetStuntReaction;
  int nTargetType;
  DynArray<AISIGNAL> vSignals;
  int secondaryIndex;
  CAIMannequinCommandList<32> mannequinRequest;
  SAICoverRequest coverRequest;
  EBodyOrientationMode bodyOrientationMode;
};
```

File

IAgent.h

SOBJECTSTATE::EFireControllerIndex Enumeration

C++

```
enum EFireControllerIndex {
  eFireControllerIndex_None = 0,
  eFireControllerIndex_First = 1,
  eFireControllerIndex_Second = 2,
  eFireControllerIndex_Third = 3,
  eFireControllerIndex_Fourth = 4,
  eFireControllerIndex_All = 17185
};
```

File

IAgent.h

Members

Members	Description
eFireControllerIndex_All = 17185	This value right shifted (4*n) times put in & with 0xf should give you the index of the n+1 fire controller index

SOBJECTSTATE::actorTargetReq Data Member

C++

```
SAIActorTargetRequest actorTargetReq;
```

SOBJECTSTATE::aimObstructed Data Member

C++

```
bool aimObstructed;
```

SOBJECTSTATE::aimTargetIsValid Data Member

C++

```
bool aimTargetIsValid;
```

SOBJECTSTATE::allowEntityClampingByAnimation Data Member

```
C++
bool allowEntityClampingByAnimation;
```

SOBJECTSTATE::allowStrafing Data Member

```
C++
bool allowStrafing;
```

Description

Whether strafing is allowed or not.

SOBJECTSTATE::bAllowLowerBodyToTurn Data Member

```
C++
bool bAllowLowerBodyToTurn;
```

Description

FR: Temporary flag to avoid the body turn while doing specific operation. It should be removed when there could be a proper procedural clip in Mannequin to lock the turn, aiming, etc.

SOBJECTSTATE::bCloseContact Data Member

```
C++
bool bCloseContact;
```

SOBJECTSTATE::bodyOrientationMode Data Member

```
C++
EBodyOrientationMode bodyOrientationMode;
```

SOBJECTSTATE::bodystate Data Member

```
C++
int bodystate;
```

SOBJECTSTATE::bReevaluate Data Member

```
C++
bool bReevaluate;
```

SOBJECTSTATE::bTargetEnabled Data Member

```
C++
bool bTargetEnabled;
```

SOBJECTSTATE::bTargetIsGroupTarget Data Member

```
C++
bool bTargetIsGroupTarget;
```

SOBJECTSTATE::continuousMotion Data Member

```
C++
bool continuousMotion;
```

SOBJECTSTATE::coverRequest Data Member

```
C++
SAICoverRequest coverRequest;
```

SOBJECTSTATE::curActorTargetFinishPos Data Member

```
C++
Vec3 curActorTargetFinishPos;
```

Description

Return value.

SOBJECTSTATE::curActorTargetPhase Data Member

```
C++
EActorTargetPhase curActorTargetPhase;
```

Description

Return value.

SOBJECTSTATE::eLookStyle Data Member

```
C++
ELookStyle eLookStyle;
```

SOBJECTSTATE::ePeakTargetID Data Member

```
C++
tAIObjectID ePeakTargetID;
```

SOBJECTSTATE::ePeakTargetThreat Data Member

```
C++
EAITargetThreat ePeakTargetThreat;
```

SOBJECTSTATE::ePeakTargetType Data Member

```
C++
EAITargetType ePeakTargetType;
```

SOBJECTSTATE::ePreviousPeakTargetID Data Member

```
C++
tAIObjectID ePreviousPeakTargetID;
```

SOBJECTSTATE::ePreviousPeakTargetThreat Data Member

```
C++
EAITargetThreat ePreviousPeakTargetThreat;
```

SOBJECTSTATE::ePreviousPeakTargetType Data Member

```
C++
EAITargetType ePreviousPeakTargetType;
```

SOBJECTSTATE::eTargetContextType Data Member

```
C++
EAITargetContextType eTargetContextType;
```

SOBJECTSTATE::eTargetID Data Member

```
C++
tAIObjectID eTargetID;
```

SOBJECTSTATE::eTargetStuntReaction Data Member

```
C++
EAITargetStuntReaction eTargetStuntReaction;
```

SOBJECTSTATE::eTargetThreat Data Member

```
C++
EAITargetThreat eTargetThreat;
```

SOBJECTSTATE::eTargetType Data Member

```
C++
EAITargetType eTargetType;
```

SOBJECTSTATE::fDesiredSpeed Data Member

```
C++
float fDesiredSpeed;
```

Description

< in m/s

SOBJECTSTATE::fDistanceFromTarget Data Member

```
C++
float fDistanceFromTarget;
```

SOBJECTSTATE::fDistanceToPathEnd Data Member

```
C++
float fDistanceToPathEnd;
```

SOBJECTSTATE::fForcedNavigationSpeed Data Member

```
C++
float fForcedNavigationSpeed;
```

SOBJECTSTATE::fire Data Member

```
C++
EAIFireState fire;
```

SOBJECTSTATE::fireMelee Data Member

```
C++
EAIFireState fireMelee;
```

SOBJECTSTATE::fireSecondary Data Member

```
C++
EAIFireState fireSecondary;
```

SOBJECTSTATE::fMovementUrgency Data Member

```
C++
float fMovementUrgency;
```

Description

AISPEED_ZERO, AISPEED_SLOW, AISPEED_WALK, AISPEED_RUN, AISPEED_SPRINT. This affects animation and thus speed only indirectly, due to it affecting the available max/min speeds.

See Also

AISPEED_ZERO, AISPEED_SLOW, AISPEED_WALK, AISPEED_RUN, AISPEED_SPRINT

SOBJECTSTATE::forceWeaponAlertness Data Member

C++

```
bool forceWeaponAlertness;
```

SOBJECTSTATE::fTargetSpeed Data Member

C++

```
float fTargetSpeed;
```

Description

< in m/s

SOBJECTSTATE::jump Data Member

C++

```
bool jump;
```

Description

Actor's movementlookingfiring related.

SOBJECTSTATE::lean Data Member

C++

```
float lean;
```

SOBJECTSTATE::mannequinRequest Data Member

C++

```
CAIMannequinCommandList<32> mannequinRequest;
```

SOBJECTSTATE::movementContext Data Member

C++

```
unsigned int movementContext;
```

Description

flags that give context to movement, which can be used by animation selection system Hint to animation system what will happen in the future. [not used in Crysis2]

SOBJECTSTATE::nTargetType Data Member

C++

```
int nTargetType;
```

SOBJECTSTATE::peekOver Data Member

C++

```
float peekOver;
```

SOBJECTSTATE::predictedCharacterStates Data Member

C++

```
SAIPredictedCharacterStates predictedCharacterStates;
```

SOBJECTSTATE::projectileInfo Data Member

C++

```
SFireCommandProjectileInfo projectileInfo;
```

SOBJECTSTATE::requestedGrenadeType Data Member

C++

```
ERequestedGrenadeType requestedGrenadeType;
```

SOBJECTSTATE::secondaryIndex Data Member

C++

```
int secondaryIndex;
```

SOBJECTSTATE::vAimTargetPos Data Member

C++

```
Vec3 vAimTargetPos;
```

Description

The requested position to aim at. This value is slightly different from the vShootTarget, and is used for visual representation to where to shoot at. Notes: can be (0,0,0) if not begin requested!

SOBJECTSTATE::vBodyTargetDir Data Member

C++

```
Vec3 vBodyTargetDir;
```

SOBJECTSTATE::vDesiredBodyDirectionAtTarget Data Member

C++

```
Vec3 vDesiredBodyDirectionAtTarget;
```

SOBJECTSTATE::vDirOffPath Data Member

C++

```
Vec3 vDirOffPath;
```

Description

displacement vector between the physical position and the current path.

SOBJECTSTATE::vForcedNavigation Data Member

C++

```
Vec3 vForcedNavigation;
```

SOBJECTSTATE::vInflectionPoint Data Member

C++

```
Vec3 vInflectionPoint;
```

Description

If non-zero, contains the estimated future move target once the agent reaches the current move target.

SOBJECTSTATE::vLobLaunchPosition Data Member

C++

```
Vec3 vLobLaunchPosition;
```

SOBJECTSTATE::vLobLaunchVelocity Data Member

C++

```
Vec3 vLobLaunchVelocity;
```

SOBJECTSTATE::vLookTargetPos Data Member

C++

```
Vec3 vLookTargetPos;
```

Description

The requested position to look at. Notes: can be (0,0,0) if not begin requested!

SOBJECTSTATE::vMoveDir Data Member

C++

```
Vec3 vMoveDir;
```

SOBJECTSTATE::vMoveTarget Data Member

C++

```
Vec3 vMoveTarget;
```

Description

If non-zero, contains the absolute position the agent should (and can) move towards.

SOBJECTSTATE::vShootTargetPos Data Member

C++

```
Vec3 vShootTargetPos;
```

Description

The requested position to shoot at. This value must be passed directly to weapon system, the AI system has decided already if the shot should miss or hit. Notes: can be (0,0,0) if not begin requested!

SOBJECTSTATE::vSignals Data Member

C++

```
DynArray<AISIGNAL> vSignals;
```

SOBJECTSTATE::vTargetPos Data Member

C++

```
Vec3 vTargetPos;
```

SOBJECTSTATE::weaponAccessories Data Member

C++

```
int weaponAccessories;
```

Description

Weapon accessories to enable.

SOBJECTSTATE::== Operator

```
C++
bool operator ==(SOBJECTSTATE& other);
```

SOBJECTSTATE::ClearSignals Method

```
C++
void ClearSignals();
```

SOBJECTSTATE::FullReset Method

```
C++
void FullReset(bool clearMoveDir = false);
```

SOBJECTSTATE::Reset Method

```
C++
void Reset(bool clearMoveDir = true);
```

SOBJECTSTATE::Serialize Method

```
C++
void Serialize(TSerialize ser);
```

SOBJECTSTATE::SOBJECTSTATE Constructor

```
C++
SOBJECTSTATE();
```

In This Topic