IA em Half-Life

Modelagem de Jogos Digitais

História

- O primeiro jogo desenvolvido pela Valve foi Half-Life, empresa fundada em 1996.
- Apresentado pela primeira vez ao público em 1997 na E3 e encantou a todos.
- Data de lançamento foi adiada várias vezes durante 1998, antes do jogo ser finalmente lançado em novembro do mesmo ano.
- Nome original do projeto do jogo era Quiver, inspirado na base militar Arrowhead, do livro de Stephen King, The Mist.

Half-Life

- **Half-Life** foi escolhido porque era sugestivo ao tema, não era clichê, e tinha um símbolo visual correspondente: a letra grega λ (lambda minúscula), que representa meia-vida na física nuclear.
- Uma mistura de terror e ação baseado na Engine de Quake.
- Doom foi uma grande influência para a maioria da equipe que trabalhou em Half-Life. Eles queriam Half-Life assustasse os jogadores, assim como Doom o fez.

Half-Life

- Você começa o jogo em um trem, com destino a Black Mesa, um complexo localizado no Novo México, EUA, onde várias pesquisas científicas de todos os gêneros são realizadas.
- Você irá encarnar Gordon Freeman, um cientista formado pelo MIT (Instituto de Tecnologia de Massachusetts), que trabalha na área de pesquisa de materiais anômalos e segue para um de seus trabalhos mais importantes de sua carreira, uma experiência com um cristal misterioso que fora levado a Black Mesa.

Half-Life

- Com alguns pequenos empecilhos no caminho, o herói finalmente chega ao local da experiência.
- Quando tudo parece estar indo bem, algo dá errado no experimento com o cristal e diversas explosões começam a ocorrer, o que acaba abrindo vários portais por todo o complexo, possibilitando a invasão de seres alienígenas de outra dimensão.

Premiações

- Na E3 1998 ele ganhou o Game Critics Awards de Melhor Jogo de PC e Melhor Jogo de Ação.
- Melhor jogo de 1999: Academia de Artes Interativas e Ciências, EUA.

Avaliações

Baseado em

3.528

Avaliações

Fonte: MetaCritic

IA em jogos

Praticamente todos os jogos utilizam em seu desenvolvimento alguma forma de Inteligência Artificial.

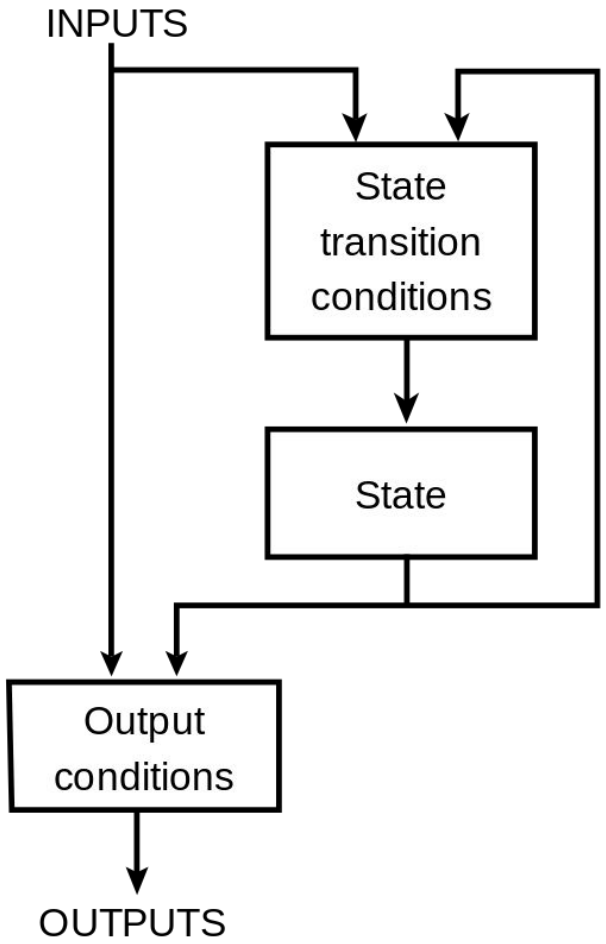
Amplamente utilizada no controle do comportamento de personagens não controlados pelo jogador, conhecidos por (nonplayer characters - NPCs).

Desenvolvimento

- O game utiliza “máquina de estado orientada por programação”, que é outra maneira de dizer que é dependente do estado e dirigida por respostas.
- Durante o desenvolvimento cerca 70% da engine de Quake teve de ser reescrita para se adaptar ao game.
- Por ser um estúdio novo, muitas dificuldades foram encontradas na época.

Desenvolvimento

- Ao colocar em camadas as FSMs, os desenvolvedores conseguiram alcançar uma complexidade notável na forma como o IA se comporta - monstros podem entrar em pânico quando fogem, se movimentam quando atacam o jogador e buscam reforços se perceberem que estão perdendo uma batalha.
- Um dos contras em utilizar FSMs é que o jogo se torna previsível;



Funcionamento de uma FSM

Máquina de Estado Finito

Funcionamento de FSMs

- **Estados:** Representam uma posição no tempo que, consequentemente irá influenciar no comportamento do agente.
- **Transições:** São ligações entre os estados.
- **Eventos:** São ações que ocorrem ao redor do agente.
- **Condições:** São regras que devem ser preenchidas para que ocorra a mudança de estado

Estrutura do jogo

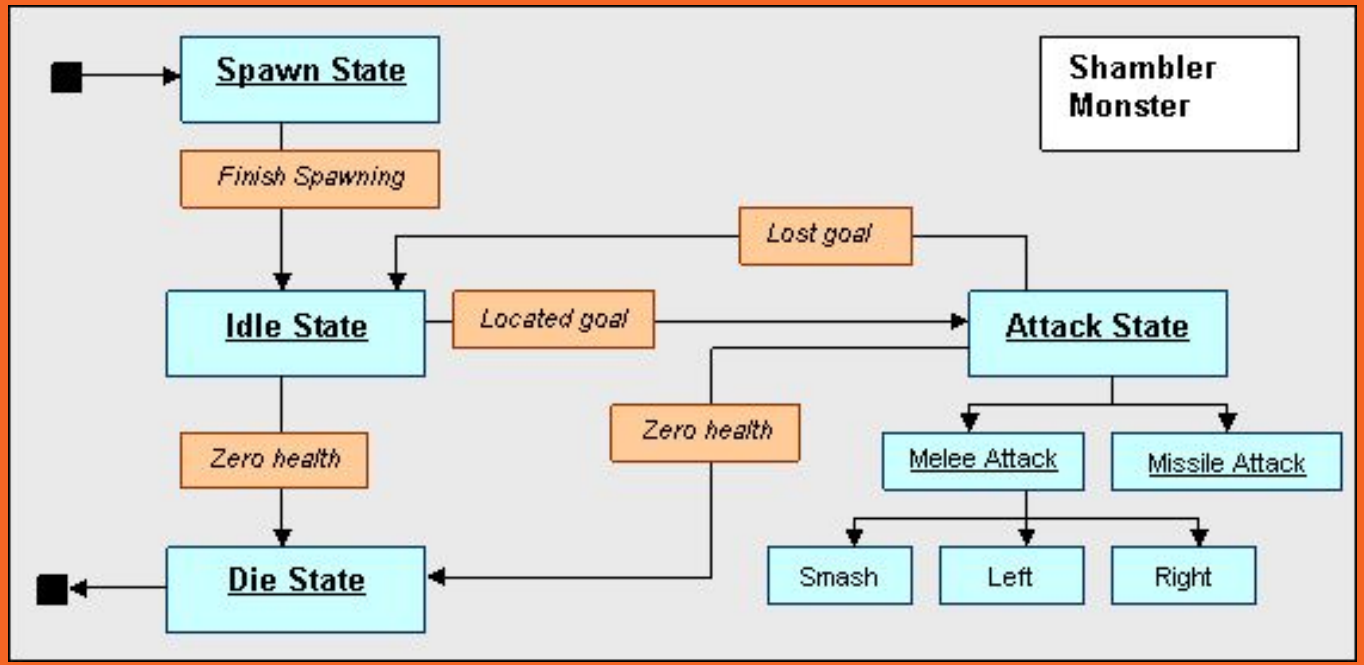
- **Agendador e Sistema de Metas:** Na programação de arquivos [h, cpp], você encontrará um sistema muito simples orientado por objetivos. Consiste em várias camadas de tarefas que podem ser combinadas processualmente.
- **Tarefas:** Tarefas são comportamentos atômicos curtos que são definidos para um propósito específico. Por exemplo, a maioria dos atores no Half-Life suporta o seguinte: TASK_WALK_PATH, TASK_CROUCH, TASK_STAND, TASK_GUARD, TASK_STEP_FORWARD, TASK_DODGE_RIGHT, TASK_FIND_COVER_FROM_ENEMY, TASK_EAT, TASK_STOP_MOVING, TASK_TURN_LEFT, TASK_REMEMBER. Eles são definidos como enumerações no arquivo de cabeçalho e implementados como métodos C++.

Estrutura do jogo

- **Condições:** são usadas para expressar a situação de um ator no mundo. Como tudo é codificado, as condições podem ser expressas muito compactamente como um bitfield, mas nesse caso ele limita as diferentes condições a 32. Por exemplo, as condições são COND_NO_AMMO_LOADED, COND_SEE_HATE, COND_SEE_FEAR, COND_SEE_DISLIKE, COND_ENEMY_OCCLUDED, COND_ENEMY_TOOFAR, COND_HEAVY_DAMAGE, COND_CAN_MELEE_ATTACK2, COND_ENEMY_FACING_ME.
- **Agendamentos:** Um cronograma é composto de uma série de tarefas (com parâmetros arbitrários) e recebe um campo de bits de condições para ajudar a especificar quando esse cronograma é inválido. O objeto de agendamento básico também tem um nome para ajudar na depuração.

Estrutura do jogo

- **Objetivos:** Em um nível mais alto, as metas são compostas por vários agendamentos. A lógica na meta pode selecionar uma programação conforme necessário com base em qual tarefa falhou e qual é o contexto atual. Os objetivos no Half-Life incluem GOAL_ATTACK_ENEMY, GOAL_MOVE, GOAL_TAKE_COVER, GOAL_MOVE_TARGET e GOAL_EAT.



Aplicação da Máquina de Estado no game

Máquina de Estados

Rodar tarefa

```
// RunAI - substituído por bullsquid
// precisa ser verificado a cada pensamento.
//=========================================================
void CBullsquid :: RunAI ( void )
{
    // primeiro, fazer coisas de classe base
    CBaseMonster :: RunAI();

    if ( pev->skin != 0 )
    {
        // fechar o olho se estiver aberto.
        pev->skin = 0;
    }

    if ( RANDOM_LONG(0,39) == 0 )
    {
        pev->skin = 1;
    }

    if ( m_hEnemy != NULL && m_Activity == ACT_RUN )
    {
         // se não houver inimigos e estiver correndo
        if ( (pev->origin - m_hEnemy->pev->origin).Length2D() < SQUID_SPRINT_DIST )
        {
            pev->framerate = 1.25;
        }
    }
```

Máquina de Estados

Rodar tarefa

```
void CScientist :: RunTask( Task_t *pTask )
{
  switch ( pTask->iTask )
  {
  case TASK_RUN_PATH_SCARED:
    if ( MovementIsComplete() )
      TaskComplete();
    if ( RANDOM_LONG(0,31) < 8 )
      Scream();
    break;

 [...]
```

Máquina de Estados

Tarefa curta

```
[...]
case TASK_HEAL:
    if ( m_fSequenceFinished )
    {
      TaskComplete();
    }
    else
    {
      if ( TargetDistance() > 90 ) // distância do alvo
        TaskComplete();
      pev->ideal_yaw = UTIL_VecToYaw( /* ... */ );
      ChangeYaw( pev->yaw_speed );
    }
    break;

  default:
    CTalkMonster::RunTask( pTask );
    break;
  }
```

Demonstração da Máquina de Estados

Fonte: Canal no Youtube: Jesus Entediado