A3PI

A3PI

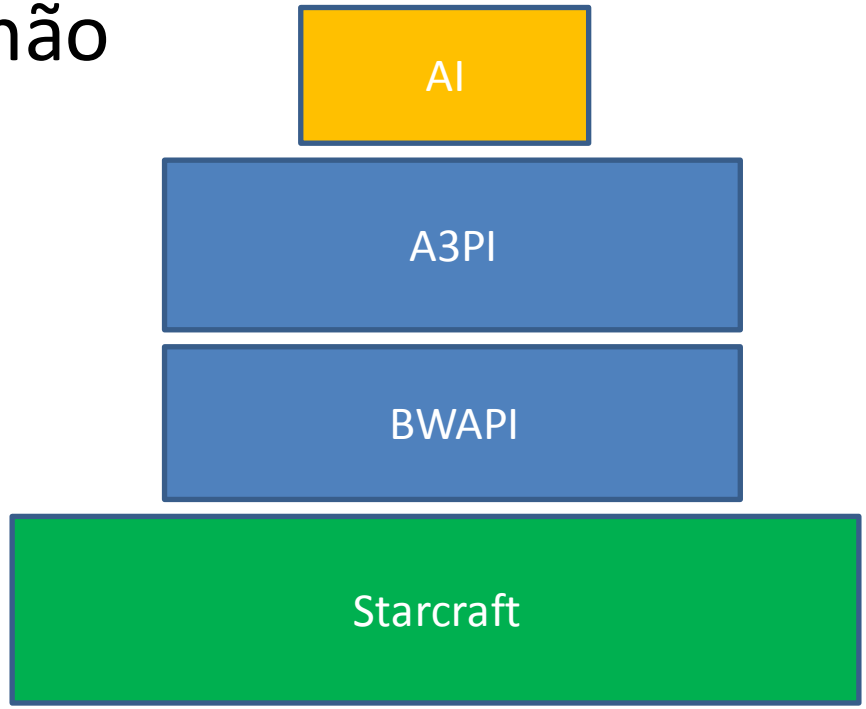
- Tem como objetivo estender o BWAPI para que a IA seja elaborada no paradigma orientado a agentes
- Nela deve-se perder a figura de um componente central que comanda todo o restante diretamente
- Todos os agentes devem tomar ação em paralelo e não sequencialmente . Por isso é interessante o uso de threads.

Porque BWAPI não é ideal para criar agentes

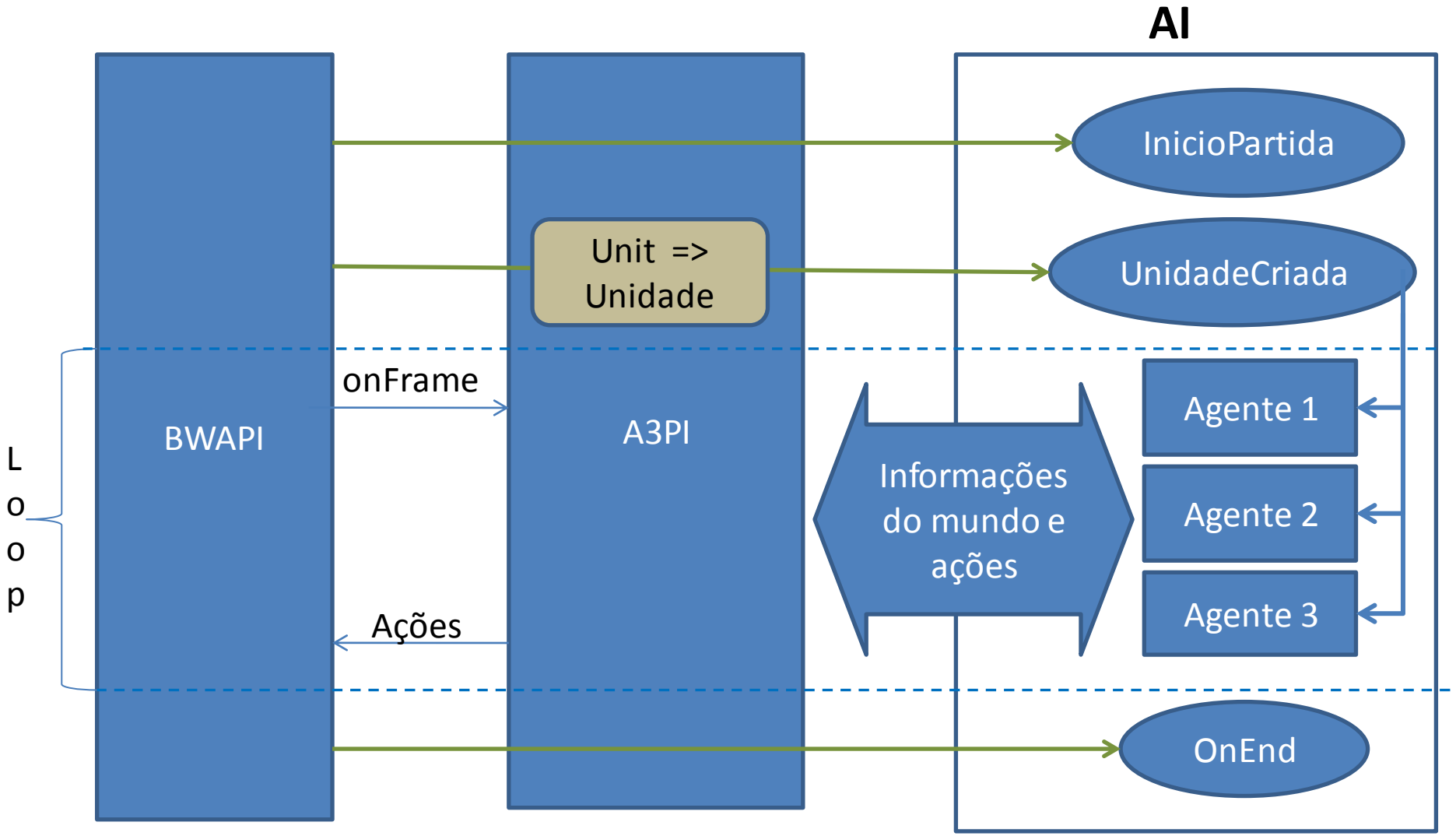
- Muitos métodos retornam informação global do mundo e não apenas da visão do agente.
- Não oferecem uma estrutura para threads serem usadas.
  - Muitos métodos causam crash se executados em paralelo
- A forma que é feito convida o programador a criar uma estrutura centralizada e sequencial.

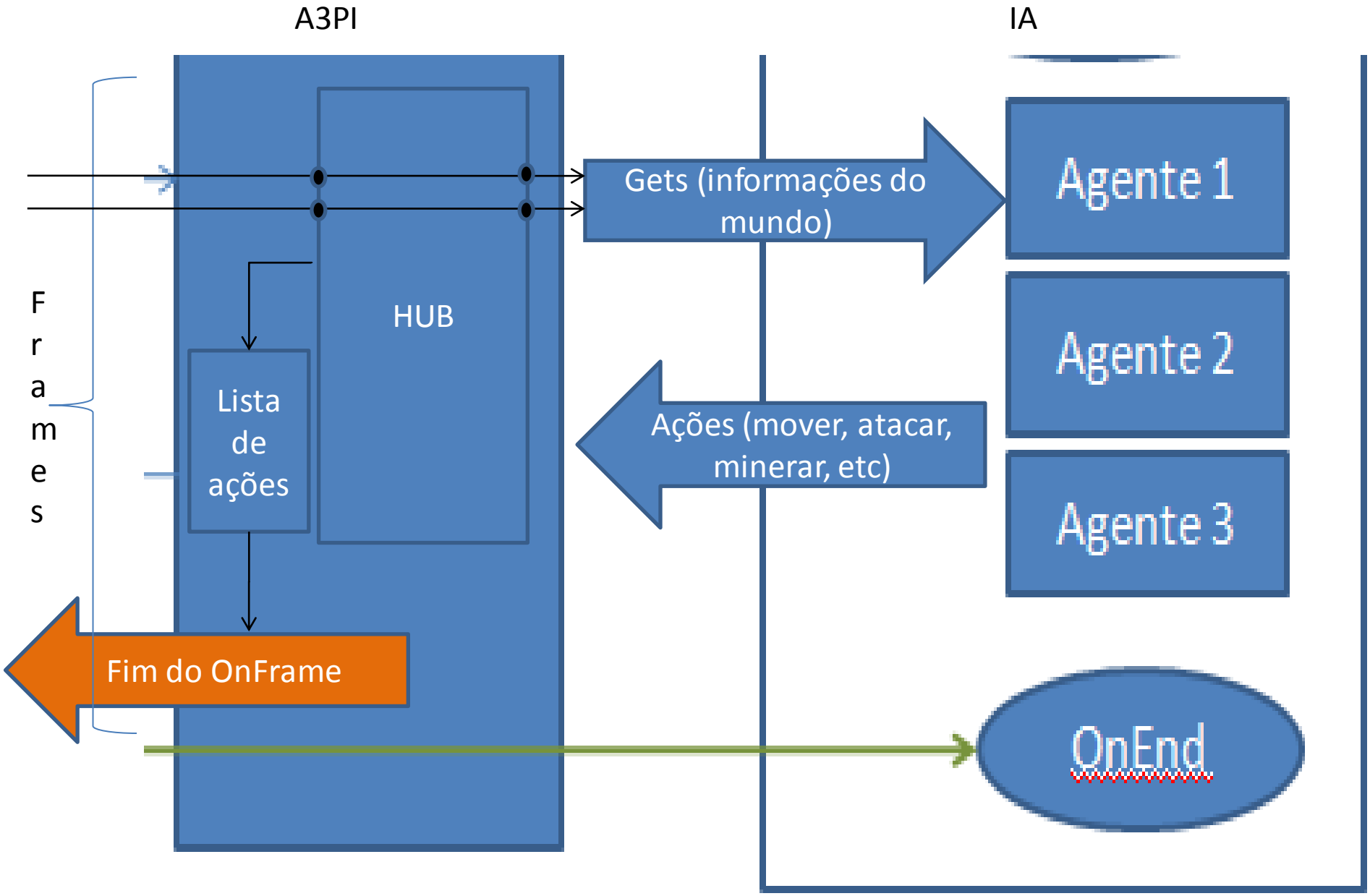
Arquitetura

- O programador passa a não “ver” mais o BWAPI
- Muitas estruturas do BWAPI foram reaproveitadas



Arquitetura




A3PI
IA
HUB
Lista de ações
Frames
Gets (informações do mundo)
Ações (mover, atacar, minerar, etc)
Agente 1
Agente 2
Agente 3
Fim do OnFrame
OnEnd

Diferenças de Unidade e Unit

- getPlayer, getId e alguns outros métodos foram retirados.
- Os métodos de ação (mover, construir, atacar, etc) são executados no final de turno, porém retornam um bool informando se poderão mesmo ser executados.
- Os métodos que requisitam informação retornam imediatamente, mas não todos são disponíveis.
- Alguns métodos foram adicionados.

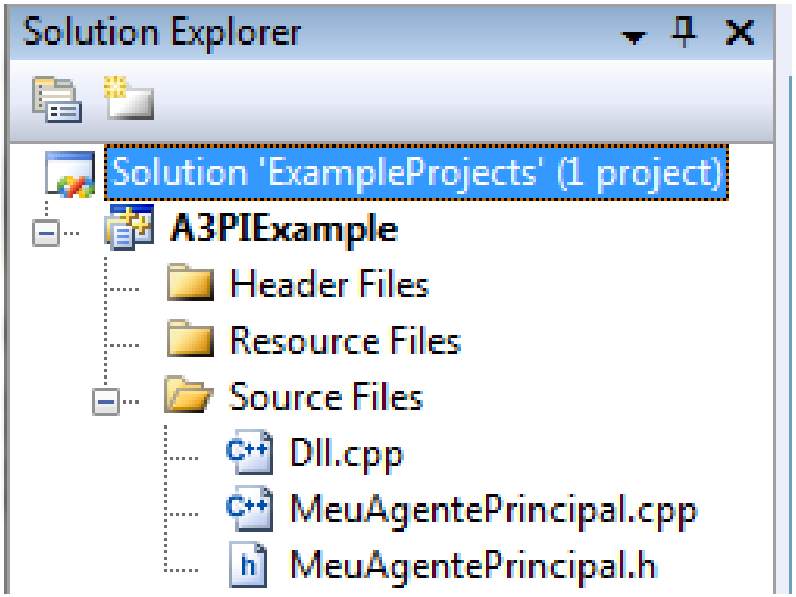
Métodos adicionados em Unidade

- getMinerals – retorna todos os minerais que a unidade consegue ver.
- getGeysers - retorna todos os geysers que a unidade consegue ver.
- getEnemyUnits – todas as unidades inimigas
- getAllyUnits – todas as unidades aliada
- isEnemy – se uma dada Unidade é inimiga
- isAlly – se uma dada Unidade é aliada
- minerals – quantidade de mineral que o time possui
- gas – quantidade de geyser que o time possui
- supplyTotal - retorna quanto supply o time possui
- supplyUsed - retorna quanto supply o time estah gastando
- checkNovoTurno – checar se o comando já foi executado.

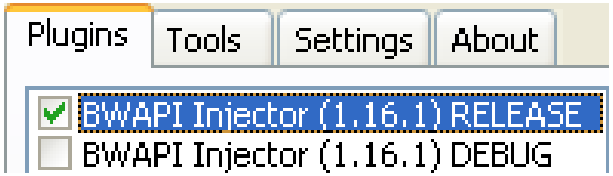
... Olhar Unidade.h!

Como usar a A3PI

- Será fornecida uma Solution do Visual Studio com um projeto de exemplo com nossa biblioteca já “linkada”
- Foi usada a última versão do BWAPI (3.7.2).
- Já está tudo pré-configurado e “mastigado”, basta baixar, deszipar e usar.

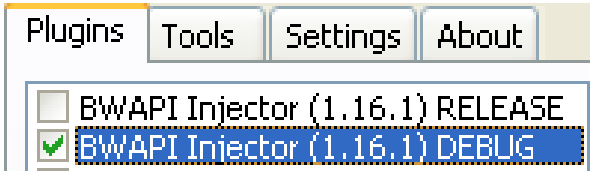


Configurações



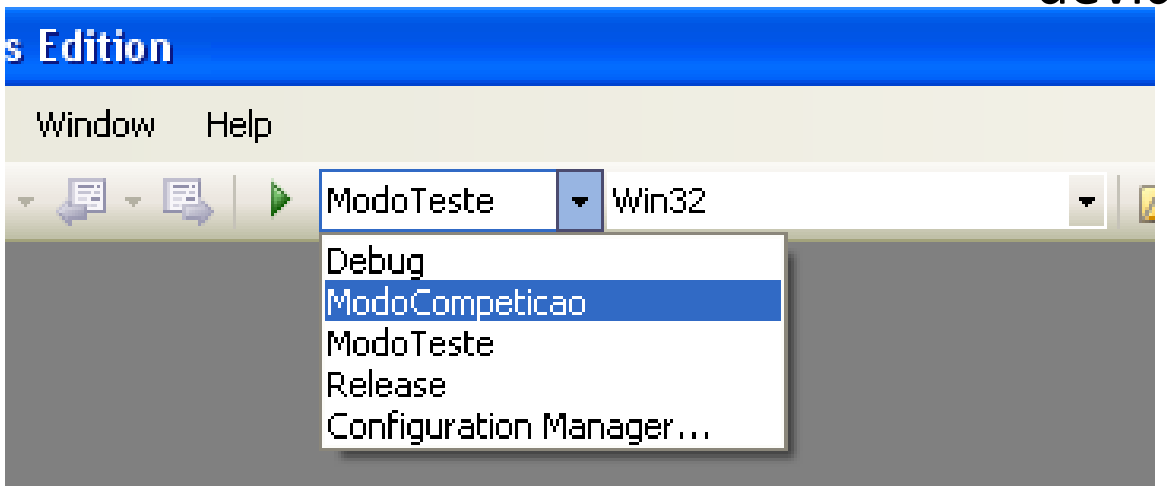
Modo Competição

- Projeto em Release
- Mais rápido e estável
- Usa Dll.
- Será utilizado na competição



Modo Teste

- Projeto em Debug
- Breakpoints e console com printf's
- Gera um .exe
- Mais lento e gera mais erros devidos ao Asserts de debug



Configurações

- Modo teste foi provido para auxiliar o desenvolvimento
- Mesmo não precisando alterar códigos, ambos os modos são implementados de forma diferente no BWAPI.
- Recomendado usar quase sempre o modo competição e excepcionalmente usar o modo teste quando precisar depurar código.

Como usar

- Implementar a classe abstrata AgentePrincipal com os métodos InicioPartida, UnidadeCriada e onEnd.

```
void MeuAgentePrincipal::UnidadeCriada(Unidade* unidade){
    //Uma nova unidade sua foi criada (isto inclui as do inicio da partida).
    //Implemente aqui como quer tratar ela.
    BWAPI::UnitType tipo = unidade->getType();

    //Nao desperdicar threads com predios que nao fazem nada
    if(tipo != BWAPI::UnitTypes::Protoss_Pylon &&
        tipo != BWAPI::UnitTypes::Protoss_Assimilator){
        CreateThread(NULL,0,threadAgente,(void*)unidade,0,NULL);
    }
    else{
        //talvez voce queira manter a referencia a estes predios em algum lugar
    }
}
```

```
DWORD WINAPI threadAgente(LPVOID param){

    Unidade *u = (Unidade*) param;
    while(true){

        if(GameOver || u == NULL || !u->exists())
            return 0;


        if(!u->isCompleted()){
            Sleep(500);
            continue;
        }
        if(!u->checkNovoTurno()){
            Sleep(10);
            continue;
        }

        if(u->isIdle()){ //nao ta fazendo nada, fazer algo util
            if(u == amigoDaVez) AIConstrutora(u);
            else if(u->getType().isWorker()) AITrabalhador(u);
            else if(u->getType().canProduce()) AICentroComando(u);
            else AIGuerreiro(u);
        }
        ...
```

```
void MeuAgentePrincipal::onEnd(bool isWinner){
    GameOver = true;
    Sleep(550);
}
```

Arquivo conf.txt

- Arquivo texto que configura a A3PI com ou sem fog of war
- Nele haverá 0 ou 1 indicando se o fog of war está ativo
- Caso esse arquivo não seja encontrado, o fog of war vai ser desabilitado