Principles of Programming, Spring 2006
Project: Quang

Daejun Park, Heejong Lee
Programming Research Lab.@SNU

Due: June 9, 2006

프롤로그

게임회사 프원은 이번에 Quang이라는 게임을 출시하려고 합니다. Quang은 두 팀이 서로 대결을 펼치는 게임인데, 각 팀의 대장이 자신의 부하 유닛을 조작하여 상대편 대장을 물리치면 이기게 됩니다. 이 게임은 과거 사용자가 직접 실시간으로 전략을 세워 행동하는 방식을 탈피하여, 스스로 전략을 세워 행동할 수 있는 프로그램끼리 게임을 벌일 수 있도록 지원합니다. 즉, 사용자가 콘솔로 직접 유닛을 조작하는 것이 아니라, 사용자가 제출한 프로그램이 게임의 시작부터 끝까지 유닛을 자동으로 조작하게 됩니다.

프원은 곧 출시될 Quang을 홍보하기 위해, 출시 3일 후에 대회를 열려고 합니다. 게임이 출시되면, 전략 프로그램의 구현을 위한 명세가 제공이 되고, 대회 참가자들은 3일이라는 제한된 짧은 시간내에, 전략을 세워서 참가하게 됩니다. 대회에서 우수한 성적을 거둔 팀에게는 부상으로 "프원의 A학점"이 제공됩니다.

그런데, "타"라는 유능한 해커가 프원의 서버에 침입하여, 현재까지 완성된 초벌 명세를 알아내어 공개를 했습니다.

초벌 명세는 완성된 Quang의 정확한 명세는 아니지만, Quang개발자들이 게임 구현의 얼개로 사용하는 명세입니다. 이 명세를 전혀 모르는 상황에서는 전략을 구현할 시간이 출시 후 3일 밖에 없었지만, 이제 초벌 명세가 공개되었기 때문에, 상위 레벨에서 전략을 미리 세워 놓는 것이 가능해졌습니다.

게임 출시 전인 지금부터, 해킹된 초벌 명세를 이용하여 상위 레벨의 전략을 세운 다음, 몇 주 뒤에 Quang의 출시와 함께 정확한 명세가 공개되면, 미리 만들어 놓은 상위 레벨의 전략에 하위 레벨의 인터페이스를 추가하여, 대회

에 참가하는 것이 유리할 것으로 보입니다. 그렇게 할 경우, 3일 만에 모든 구현을 한 참가자보다 우수한 성적을 내는 것이 쉽기 때문입니다. 하지만 하위 레벨의 인터페이스를 추가하는 작업을 3일만에 완벽하게 해내는 것이 그리 간단한 일이 아니기 때문에, 상위레벨의 전략을 세울 때, data abstraction을 잘 고려하여 프로그램을 구현해야 할 것입니다. 또한 최근 소식에 의하면, 프원이 해킹을 당한 후에, 여러분처럼 미리 전략을 구현해 놓을 참가자들을 의식한 나머지, Quang의 내부 구현을 약간 변경할 계획이라고 합니다. 따라서 data abstraction의 개념이 그 어느때보다도 중요하다고 할 수 있습니다.

초벌 명세

전략 프로그램에 구현되어야 하는 함수는 다음과 같습니다.

$$\begin{aligned} \texttt{move} &: \mathit{PBoard} \to (\mathit{Pos} \times \mathit{Dir} \times \mathit{Dir})\ \texttt{list} \\ \texttt{init} &: \mathit{unit} \to \mathit{Info}\ \texttt{list} \end{aligned}$$

`move`는 전체 보드판의 특정 부분의 정보만 받아서, 유닛의 이동 정보를 리턴하는 함수입니다. `init`은 자신의 유닛을 보드판의 어느 곳에 놓을지 초기화하는 함수 입니다. 위 함수들의 타입정의에 사용된 도메인은 다음과 같습니다.

$$\begin{aligned} \mathit{PBoard} &= \mathit{Pos} \times \mathit{State} \\ \mathit{State} &= \mathit{Hp} \times \mathit{RangeAttr} \times \mathit{AmmoAttr} \times \mathit{IsCaptin} \times \mathit{Id} \\ \mathit{Info} &= \mathit{Pos} \times \mathit{RangeAttr} \times \mathit{AmmoAttr} \times \mathit{IsCaptin} \end{aligned}$$

$$\begin{aligned} \mathit{Pos} &= \mathbb{N} \times \mathbb{N} \\ \mathit{Dir} &= \mathit{Up} \mid \mathit{Down} \mid \mathit{Left} \mid \mathit{Right} \\ \mathit{RangeAttr} &= \mathit{Short} \mid \mathit{Medium} \mid \mathit{Long} \\ \mathit{AmmoAttr} &= \mathit{Light} \mid \mathit{Heavy} \\ \mathit{IsCaptin} &= \mathit{True} \mid \mathit{False} \end{aligned}$$

Quang은 사용자가 제출한 전략 프로그램을 이용해 다음과 같이 전체 게임을 진행합니다.

$$\texttt{A.init} \to \texttt{B.init} \to \texttt{A.move} \leftrightarrows \texttt{B.move}$$

게임이 시작하면, Quang은 A팀과 B팀의 `init`함수를 호출하여 A팀과 B팀의 초기화를 각각 진행합니다. 초기화가 끝난 후, 보드판에 각 팀의 유닛 셋팅을 보여줍니다. 그 후, A팀과 B팀이 번갈아 가면서 각자의 턴을 가지게 됩니다. 한 턴에 할 수 있는 유일한 일은 자신의 유닛에 대해 이동명령을 내리는 것입

니다. 이 때, 대장의 시야가 확보되는 범위 만큼의 보드판 정보가 주어집니다. 명령이 내려지고 난 후, 커맨드센터는 그 팀의 유닛을 이동한 후, 자동으로 공격을 수행시킵니다. 공격 후, 상대팀의 대장이 죽었으면 게임을 끝내고, 그렇지 않을 경우, 상대팀으로 턴이 넘어가게 됩니다. 일정 턴 이상이 지나도 게임이 끝나지 않으면, 현재 남아 있는 유닛의 체력의 합계가 큰 팀이 이기게 됩니다.

공격시에 유닛은 자신의 공격 범위 안에 있는 모든 상대편 유닛을 공격합니다. 이 때, 공격은 명중률에 따라서 성공할수도 있고, 실패할 수도 있습니다. 공격이 성공하게 되면, 공격 유닛과 방식에 따라 데미지가 결정됩니다. 그리고 데미지 만큼의 체력이 감소하게 됩니다. 체력이 0이 되면 죽게됩니다.

공격 범위는 그 유닛의 *RangeAttr*에 의해 결정됩니다. *Short*는 1칸에서 10칸, *Medium*은 11칸에서 30칸, *Long*은 31칸에서 60칸의 공격범위를 가짐을 의미합니다. 명중률은 공격 유닛의 *RangeAttr* factor와 *AmmoAttr* factor를 곱한 값입니다. ($Short = 0.8$, $Medium = 0.5$, $Long = 0.2$, $Heavy = 0.5$, $Light = 1.0$) 유닛이 *LightAmmoAttr*를 가진 유닛에게 공격을 받았을 때, 기본 데미지는 30, *HeavyAmmoAttr*를 가진 유닛에게 공격을 받았을 때 50이고, 측면에서 공격을 받았을 경우 1.5배로 늘어나며, 뒤에서 공격을 받았을 경우에는 2배로 데미지가 늘어나게 됩니다.

유닛의 이동은 한번에 한칸씩만 가능하며, 이동후에는 어떠한 방향으로 서 있을지 결정해야 합니다. 전체 유닛은 대장을 포함하여 10개이고, 각 유닛의 기본 체력은 100입니다.