CORBA
Dyna Blaster

Łukasz Chojnacki Daniel Zieliński

6 lutego 2009

1 Rozmyślania przedimplementacyjne

Początkowo pojawiało się kilka pomysłów jak podejść do sprawy implementacji logiki gracza w grze „*Dyna Blaster*". Rozpatrywaliśmy próby zaimplementowania algorytmu genetycznego (chociażby w oparciu o bibliotekę *JGAP - http://jgap.sourceforge.net/*), jednak ze względu na szybkość rozgrywki ostatecznie postawiliśmy na proceduralne implementowanie gracza i dokładanie kolejnych składowych wpływających na jego zachowanie.

2 Sztuczna inteligencja gracza

Kontroler sterujący graczem rozpatruje każdą przychodzącą ramkę i na bieżąco aktualizuje informacje niezbędne do wykonania w danej chwili najlepszego ruchu.
Procedura:

- *lookupCollectedRangeBonuses* steruje informacjami na temat zasięgów bomb przeciwników. Przy uruchomieniu kontrolera, dla wszystkich aktualnie zarejestrowanych graczy ustawiany jest domyślny zasięg bomby, zdefiniowany w klasie *Globals*. W momencie kiedy dany gracz zdobędzie bonus powiększający zasięg bomby, informacje na temat zasięgu danego gracza są aktualizowane. Jeśli więcej niż jeden gracz stoi w danej chwili na polu, na którym wcześniej stał bonus, to dla bezpieczeństwa wszystkim ustawiany jest zasięg na większy. Podobnie zasięgi są aktualizowane w momencie, kiedy wykryto, że dany gracz zginął lub gdy nowy gracz dołączył do gry;

- *createBoardModel* wyznacza rozmieszczenie bomb i przybliżone czasy wybuchów, co pozwala wyznaczyć drogę ucieczki w sytuacji, w której gracz znajduje się na polach niebezpiecznych. Procedura uwzględnia również rekurencyjne wybuchy i jeśli zasięgi się nakładają to dla bomb występujących w łańcuchu ustawiany jest czas wybuchu tej bomby, która wybuchnie najwcześniej;

- *canPlaceBomb* sprawdza czy w danej chwili gracz może postawić bombę. Zakłada ona, że w następnej ramce każdy z graczy postawi bombę. Jeśli w takim przypadku uda się graczowi uciec w bezpieczne miejsce a przeciwnicy znajdują się w jego otoczeniu, to gracz stawia bombę;
- *doOffensiveMove* sprawdza dystans do najbliższego bonusu/gracza i kieruje zawodnika w tę stronę (przy wykorzystaniu algorytmu przeszukiwania wszerz);
- *escapeMove* uruchamiana jest w przypadku gdy gracz znajduje się w zasięgu jakiejś bomby, czyli na polu niebezpiecznym. Algorytm przeszukiwania wszerz znajduje najbliżej leżące bezpieczne pole.

3 Turniej

W turnieju udało nam się uzyskać trzecie miejsce ex aequo z botem *kazik1616* — pokonaliśmy *Rabbita* jednak w drodze do finału ulegliśmy *Brutalowi*. Podczas turnieju dało się zauważyć, że gracz nie zawsze gra rozsądnie. Udało nam się doprowadzić do sytuacji, w której gracz jest bardziej agresywny — podczas turnieju ciekawszym zajęciem dla bota było niszczenie murków i zbieranie bonusów niż atakowanie przeciwnika. Jednak ze względu na trudność debugowania nie udało nam się usunąć wszystkich mankamentów. Zdarzają się czasem sytuacje, w których gracz postawi sobie sam bombę i nie da rady uciec.

4 Informacje dodatkowe

Zaimplementowane zostały dwa interfejsy:

- *IPlayerFactory*, klasa **com.lcdz.dyna.factories.AIPlayerFactory**
- *ICPlayerFactory*, klasa **com.lcdz.dyna.factories.AICPlayerFactory**

Aby zbudować archiwum jar zawierające obie implementacje, należy uruchomić zadanie „*jar*" znajdujące się w załączonym skrypcie ant'owym ustawiając wcześniej w pliku *build.properties* scieżki do projektu dyna-corba oraz dyna-core (wymagane zależności, żeby skompilować kod gracza).