**Risk Game AI DLL API**

*First Created January 8, 2014*

The Risk Game has an API for installing more Artificial Intelligence engines to not only expand the capabilities of the game, but also to allow hacking by the user. This will expand the amount of enjoyment beyond just sitting in front of the computer veging out while mindlessly playing a game. You will be able to create your own AI and pit it against the existing AIs as well as your self.

All of the API calls use standard C calling conventions and naming. Sorry, no C++ interfacing. I do this so that any C compiler may be used to write the code. Naming conventions for C++ vary widely not only from vendor to vendor, but also even version to version. The compiler I used for this project is Visual C++ 6.0. This is a little long in the tooth, since this is the compiler that was released for use with Windows NT and Windows 98. Using standard C naming means that you can even use gcc if you wish to write your dll.

All export functions should be declared as extern “C” if you are using C++ to write the code. If you look in my files, you will see that this is what I did.

**Interface Functions**

**void \*GetParams(void);**

This function is used to get a pointer to the default parameters for the AI evaluation functions. It is these parameters that are used to adjust the weighting of all the things that the evaluation function uses to make its decisions. It is assumed that the default values are the best that they can be, but this may not be true. For example, when you first write the AI, you have no idea really just exactly what these values should be. You may have a pretty good idea, but there may be some optimal value that may surprise even you.

**int GetParamSize(void)**

This function is used to get the size in bytes of the parameter of interest.

**int GetParamIndex(int ParamNumber);**

This function is used to the get byte index of the nth parameter.

**int GetTotalParams(void)**

This function returns the total number of parameters in the parameter table.

**int SetParameter(void \*pParams, int ParamNumber, long value);**

This function is used to set a parameter in the parameter table. The values are always passed as longs, but they may, or may not, be truncated by the DLL depending of the size of the parameter.

**void EditParamDLG(void \*pParams);**

This function is used to start up a Dialog Box for editing the parameters for the AI. The parameter is a pointer to the parameter block you want to edit. If the OK button is clicked, then the parameter block is updated.

**CMapTerritory \*Attack(void \*pParam, CMap \*pM, CPlayer \*pPlayer);**

This function is used to select a territory to attack. If the function decides that it is a bad idea to attack, it returns NULL. Parameter pParam is a pointer to the parameter block that sets the weighting for the decisions, pM is a pointer to the game map, and pPlayer is a pointer to the player whose turn it is.

**CMapTerritory \*AttackFrom(void \*pParam, CMap \*pM, CPlayer \*pPlayer,CMapTerritory \*pAT);**

This function is used to figure out from which territory the attack is to be carried out. If this function returns NULL, that indicates that the attack is to be called off. In other words, the AI has chickened out. This function is called after each iteration of the attack, so the AI can decide to switch territories to attack from, as well as cancelling the attack.

Parameters:

pParam…..pointer to the parameter block to use for the attack.

pM pointer to the game map.

pPlayer pointer to the player whose turn it is

pAT pointer to the territory being attacked

return value

NULL is attack is called off

Otherwise a pointer to the territory to attack from.

**CMapTerritory \*Choose(void \*pParam, CMap \*pM);**

This function is used to select a territory during the game setup. The player only gets to select one territory, so no cheating.

Parameters:

pParam pointer to parameter block used to make choice

pM pointer to the game map

returns:

A pointer to the chosen territory.

**int Place(void \*pParam, CMap \*pM, CPlayer \*pPlayer);**

This fuction is used to place an asset onto a territory.

Parameters:

pParam pointer to parameter block for AI

pM pointer to the game map

pPlayer pointer to player whose turn it is

Return Value:

0 if there was nothing to place

1 on success

**int Cards(void \*pParam, CMap \*pM, CPlayer \*pPlayer);**

This function is used to determine if risk cards are going to be turned in for a bonus.

Parameters:

pParam pointer to the parameter block for this AI

pM pointer to the game map

pPlayer pointer tho player whose turn it is

Return Value:

TBD

**int FreeMove(void \*pParam, CMap \*pM,CPlayer \*pPlayer);**

This function is used to determine how assets are to be moved around.

Parameters:

pParam parameter block fo rhtis AI

pM pointer to the game map

pPlayer pointer to the player whose turn it is

Return Value:

TBD

**char \*name(void);**

This function very simply just returns the name of the AI. This is so that it can be recognized in the game setup dialogs. It returns a pointer to the name.

**int War(void \*pParam, CMapTerritory \*pA,CMapTerritory \*pD, CMap \*pM);**

This is one of the more complicated functions. There needs to be a callback so that the defender gets a chance to give its input on how the battle to to take place. The defender can even decide to chicken out and retreat. And there needs to be a callback to a “rule” function that actually conducts the battle and determines the outcome.

Parameters:

pParam pointer to the AI parameter block

pA Attacking Territory

pD Defending Territory

pM Game Map

int (\*DefCallBack)(CMapTerritory \*pA, CMapTerritory \*pD) Callback to get defenders

int (\*RuleCallBack)( CMapTerritory \*pA, CMapTerritory \*pD)

decision

int (\*BattleCallBack)(CMapTerritory \*pA, CMapTerritory \*pD) Callback to wage battle form rules plugin.

**int Defend(void \*pParam, CMap \*pM, CMapTerritory \*pT, CMapTerritory \*pA);**

This function is called when the AI needs to defend itself against aggression. The defender can decide on the type of defense to mount, if any.

Parameters:

pParam pointer to the parameter block for the AI

pM pointer to the game map

pT Territory to defend

pA Attacking Territory

Return Value

Returns 0 on retreat

True otherwise

**Risk Game Rules Plugin DLL API**

We also can modify the game by writing our own rules for the game. We want all of the players to follow the same rules. Now, it is possible for the AI to over rule the rules, but that is also called cheating, so lets not go there.

Among the things that the rules plugin does is determine bonus assets, Risk Card bonus, Battle outcome.

**int Bonus(CMap \*pMap, CPlayer \*pPlayer);**

This function determines the bonus that the player is going to get for his current situation.

**int TurnInCards(CList \*pDeck, CCard \*pC1, CCard \*pC2, CCard \*pC3, CPlayer \*pPlayer);**

This function is used to determine the bonus a player.

**int Battle(Cplayer \*pPattack, CPlayer \*pPdefend, CMapTerritory \*pTa, CMapTerritorty \*pTd);**

This function is used to determine the outcome of a battle. I still need a way of getting the assets that the attacker and defender want to use to wage the battle.

**int Place(CPlayer \*pP, CMap \*pM, CMapTerritory \*pT, void \*asset);**

This function is used to determine if a player can place an asset into a given territory.