ZMLP v1.23

A program to change or creates a new Counter Strike mapcycle file.

This is a command line program so it must be executed from a command prompt or a batch file. Please note that this is NOT a DOS program, it is a true 32-bit program and must run under a win32 based system!

The program must be executed from the cstrike directory.

The maximum map name length is 50 characters and the maximum number of installed maps that the program supports are 500.

Special thanks to Per Larsson and Magnus Blixt for there help to test the program under it's development and for there new ideas for the program.

Syntax

zmlp [-r] [-a] [-a2] [-a3:Num] [-p] [-p2] [-p3:Num]

-r	Creates a new random mapcycle.txt from the existing mapcycle.txt file.
-a	Creates a new mapcycle with all maps that are installed and saves it into a new file named mapcycle.all. If mapcycle.all already exists it while be overwritten.
-a2	Same as –a but the new mapcycle while be saved as mapcycle.txt. The old mapcycle.txt while be overwritten.
-a3:Num	Takes Num number of random maps from the total number of installed maps and saves it into a new mapcycle.txt. Num can be a number between 1 and 500.
-p	Creates a new mapcycle with all PodBot supported maps that are installed and saves it into a new file named mapcycle.pod. If mapcycle.pod already exists it while be overwritten.
-p2	Same as –p but the new mapcycle while be saved as mapcycle.txt. The old mapcycle.txt while be overwritten.
-p3:Num	Takes Num number of random PodBot supported maps from the total number of installed maps and saves it into a new mapcycle.txt. Num can be a number between 1 and 500.

If no argument is given a short help text while be displayed in the command window.

Please note that the program while execute the arguments from the left to the right, for example the command "zmlp –r –a2" is the same as "zmlp –a2" because –a2 while overwrite the random mapcycled.txt that –r created. Please look at the third example how to create a new random mapcycle.txt with all installed maps.

Made by Samuel 'Zimpan' Jansson

Program Examples

To display the help text.

Zmlp

To create a new random mapcycle.txt.

Zmlp –r

To create a new random mapcycle.txt with all installed maps.

Zmpl –a2 -r

To create a new random mapcycle.txt with 15 PodBot supported maps.

Zmlp –p3:15

Revision history

v1.23 The .bsp bug is fixed (again).

v1.22 The –p3:Num bug is now fixed, didn't check for waypoints files.

v1.21 The problem with –a3:Num and –p3:Num (ignored the Num number) is fixed.

v1.20 Two new arguments are available (–a3:Num and –p3:Num).

v1.10 The first attempt to introduce the –a3 and –p3 arguments, didn't work!

v1.01 Help text corrected and some bug fixes.

v1.00 The first released version, the arguments –r, –a, –a2, –p and –p2 are included.

v0.03 Beta version that includes –r, -a and –a2 and a not fully working –p argument.

v0.01 Beta version that includes the –r argument.

Made by Samuel 'Zimpan' Jansson

Source code

```
//    ZMLP v1.23
//    Programmer: Samuel 'Zimpan' Jansson
//    Last compiled: 2001-06-15
//    (C)IAG Software Development 2001

#include <conio.h>
#include <fstream.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
#include <windows.h>

// Member functions
int CheckArg(char * TestStr);
void Random(void);
int All(char aOra2[5]);
int PodBot(char pOrp2[5]);
int NumOfMaps(char *Arg);

int main(int argc, char *argv[], char *envp[])
{
     int NumOfArg, i, Error=0, nRetCode = 0, FunctionError;

     // Number of arguments
     NumOfArg = argc-1;

     // Checks the arguments
     if (NumOfArg > 0) {
          for(i=1; i <= NumOfArg; i++)
            Error += CheckArg(argv[i]);
     }
     // Print help = no arguments
     else {
          cout << "ZMLP v1.23  (C)IAG Software Development" << endl << endl;
          cout << "Change or creates a new CS mapcycle." << endl << endl;
          cout << "zmlp [-r] [-a] [-a2] [-a3:Num] [-p] [-p2] [-p3:Num]" << endl << endl;
          cout << "-r\tCreates a new random mapcycle from the existing mapcycle." << endl;
          cout << "-a\tCreates a new mapcycle with all maps that are installed\r"
             << "\n  \tand saves it into a new file named mapcycle.all." << endl ;
          cout << "-a2\tCreates a new mapcycle with all maps that are installed\r"
             << "\n  \tand replace the old mapcycle.txt." << endl ;
          cout << "-a3:Num\tTakes Num number of random maps from the total number\r"
             << "\n  \tof installed maps and saves it into a new mapcycle.txt." << endl;
          cout << "-p\tCreates a new mapcycle with all PodBot supported maps and\r"
             << "\n  \tsaves it into a new file named mapcycle.pod." << endl ;
          cout << "-p2\tCreates a new mapcycle with all PodBot supported maps and\r"
             << "\n  \treplace the old mapcycle.txt." << endl;
          cout << "-p3:Num\tTakes Num number of random PodBot supported maps from the total\r"
             << "\n  \tnumber of installed maps and saves it into a new mapcycle.txt." << endl << endl;
          cout << "  \tMaximum number of entry's in the mapcycle file is 500\r\n  \tand each line can't
be longer than 50 character's." << endl << endl;
          cout << "This program can only be executed from the cstrike directory.";
          return nRetCode;
     }

     // No errors, proces arguments.
     if (Error == 0) {
```

```
        for(i=1; i <= NumOfArg; i++) {

          // If argv is -r call the Random function
          if (strcmp(argv[i],"-r") == 0)
               Random();

          // If argv is -a call the All(".all") function
          if (strcmp(argv[i],"-a") == 0) {
               FunctionError = All(".all");
               if (FunctionError == 0) {
                    cout << endl << "Error found during -a decoding!" << endl;
                    return nRetCode;
               }
          }

          // If argv is -a2 call the All(".txt") function
          if (strcmp(argv[i],"-a2") == 0) {
               FunctionError = All(".txt");
               if (FunctionError == 0) {
                    cout << endl << "Error found during -a2 decoding!" << endl;
                    return nRetCode;
               }
          }

          // If argv is -p call the PodBot function
          if (strcmp(argv[i],"-p") == 0) {
               FunctionError = PodBot(".pod");
               if (FunctionError == 0) {
                    cout << endl << "Error found during -p decoding!" << endl;
                    return nRetCode;
               }
          }
          // If argv is -p2 call the PodBot function
          if (strcmp(argv[i],"-p2") == 0) {
               FunctionError = PodBot(".txt");
               if (FunctionError == 0) {
                    cout << endl << "Error found during -p2 decoding!" << endl;
                    return nRetCode;
               }
          }
          // If argv is -a3 call the NumOfMaps function
          if (strncmp(argv[i],"-a3",3) == 0) {
               FunctionError = NumOfMaps(argv[i]);
               if (FunctionError == 0) {
                    cout << endl << "Error found during -a3 decoding!" << endl;
                    return nRetCode;
               }
          }
          // If argv is -p3 call the NumOfMaps function
          if (strncmp(argv[i],"-p3", 3) == 0) {
               FunctionError = NumOfMaps(argv[i]);
               if (FunctionError == 0) {
                    cout << endl << "Error found during -p3 decoding!" << endl;
                    return nRetCode;
               }
          }
        }
}
// Errors detected!
else {
```

```
            cout << "Error then reading argument's!!" << endl << endl << "Number of errors " << Error
<< endl;
            return nRetCode;
      }
      return nRetCode;
}


// Check arguments function
int CheckArg(char * TestStr) {
      char TestCriteria[7][15];
      strcpy(TestCriteria[0],"-r");
      strcpy(TestCriteria[1],"-a");
      strcpy(TestCriteria[2],"-a2");
      strcpy(TestCriteria[3],"-p");
      strcpy(TestCriteria[4],"-p2");
      strcpy(TestCriteria[5],"-a3");
      strcpy(TestCriteria[6],"-p3");

      int k,ArgLength,j;

      for (k=0; k<7; k++) {
            if (strncmp(TestCriteria[k],TestStr, 3) == 0) {
               if ( k > 4) {
                     ArgLength = strlen(TestStr);
                     if (ArgLength > 4 && ArgLength < 8) {
                           for (j=4; j<ArgLength; j++) {
                              if((int)TestStr[j]<48 || (int)TestStr[j]>57) {
                                 cout << "Argument " << TestStr << " is NOT supported!" << endl << endl;

                                 return 1;
                              }
                           }
                           return 0;
                     }
               }
               else
                     return 0;
              }
      }

      cout << "Argument " << TestStr << " is NOT supported!" << endl << endl;

      return 1;
}

// Creates a new random mapcycle
void Random(void) {
      char ReadBuffer[500][50]; // Worst case
      int NumOfLines=0, RandomNum, i=0, k, CheckSame, NewOrder[500];

      // Random numbers
      srand(time(0));

      // Open file
      ifstream fin;

      fin.open("mapcycle.txt");

      // Reads one line at the time from mapcycle.txt
```

```
	while (fin.getline(ReadBuffer[NumOfLines],50))
		if (strlen(ReadBuffer[NumOfLines]) > 0)
		  NumOfLines++;

	// Close file
	fin.close();

	if (NumOfLines == 0) {
		cout << endl << "Can't find mapcycle.txt or it's empty!" << endl;
		cout << endl << "Are zmlp.exe in the cstrike directory?" << endl;
		cout << endl << "Error found during -r decoding!" << endl;
	}
	else {

		// Creats a new mapcycle order
		do {
		  CheckSame = 0;

		  // Creats a random number
		  RandomNum = rand() % NumOfLines;

		  // Already in the list?
		  for (k=0; k<i; k++) {
		       if (NewOrder[k] == RandomNum)
		            CheckSame++;
		  }

		  // If not save the random value and increase i
		  if (CheckSame == 0) {
		       NewOrder[i] = RandomNum;
		       i++;
		  }

		} while (i < NumOfLines);

		// Save the new file in the random order
		ofstream fout ("mapcycle.txt");
		for (k=0; k < NumOfLines; k++)
		  fout << ReadBuffer[NewOrder[k]] << endl;
		fout.close();
	}
}

// Creates a new mapcycle with all maps
int All(char aOra2[5]) {
	char CurDir[256], MapDir[256], MapsFound[500][50], SaveFileName[13];
	WIN32_FIND_DATA FindData;
	HANDLE hFind;
	int AllFiles = 0, NumOfMaps = 0;

	// Get current directory and save's the string into CurDir
	GetCurrentDirectory(256,CurDir);

	// Copy CurDir to MapDir and adds \maps into the end
	strcpy(MapDir,CurDir);
	strcat(MapDir,"\\maps");

	// Change diretory to the map diretory
	SetCurrentDirectory(MapDir);
```

```
    // Creates a handle and get the first map name
    hFind = FindFirstFile("*.bsp", &FindData);

    // If no files there found print a error mesage
    if (hFind == INVALID_HANDLE_VALUE) {
            cout << endl << "Invalid File Handle. No files found!" << endl;
            return 0;
    }

    // No errors, get the rest of the map names
   else {
            // Save the first map name into the array
            strcpy(MapsFound[NumOfMaps], FindData.cFileName);
            NumOfMaps++;
            do {
              // Get the next file name
              AllFiles = FindNextFile(hFind, &FindData);
              if (AllFiles != 0) {
                   strcpy(MapsFound[NumOfMaps], FindData.cFileName);
                   NumOfMaps++;
              }
            } while (AllFiles != 0); // All maps found?

            // Close the handle
            FindClose(hFind);
    }

    // Change directory to default
    SetCurrentDirectory(CurDir);

    for (AllFiles=0; AllFiles < NumOfMaps; AllFiles++) {
            k=strlen(MapsFound[AllFiles]);
            MapsFound[AllFiles][k-4]='\0';
    }

    // If some maps there found save them into a file
    strcpy(SaveFileName,"mapcycle");
    strcat(SaveFileName,aOra2);

    if (NumOfMaps > 0) {
            ofstream fout (SaveFileName);
            for (AllFiles=0; AllFiles < NumOfMaps; AllFiles++)
              fout << MapsFound[AllFiles] << endl;
            fout.close();
    }
    return 1;
}

// Creates a new mapplist with all PodBot supported maps
int PodBot(char pOrp2[5]) {
    char CurDir[256], PodDir[256], ReadBuffer[150], WayPointDir[50]={'\0'};
    char WayPointsFound[500][50], MapsFound[500][50],Result[500][50], SaveFileName[13];
    int i, k = 0 , j = 0, j1, j2, Length;
    WIN32_FIND_DATA FindData;
    HANDLE hFind;

    // Get current directory and save's the string into CurDir
    GetCurrentDirectory(256,CurDir);

    // Copy CurDir to PodDir and adds \PodBot into the end
```

```
strcpy(PodDir,CurDir);
strcat(PodDir,"\\PodBot");

// Change diretory to the PodBot diretory
SetCurrentDirectory(PodDir);

// Creates a handle and seeks for the PodBot.cfg file
hFind = FindFirstFile("PodBot.cfg", &FindData);

if (hFind == INVALID_HANDLE_VALUE) {
        cout << endl << "Invalid File Handle. PodBot.cfg not found!" << endl;
        return 0;
}

// Close the handle
FindClose(hFind);

// Open file
ifstream fin;

fin.open("PodBot.cfg");

// Source for the "wptfolder wptcs10" line in PodBot.cfg
while (fin.getline(ReadBuffer,150))
        if (strncmp(ReadBuffer,"wptfolder",9) == 0)
          strcpy(WayPointDir,ReadBuffer);

// Close file
fin.close();

// If no directory found print error message
if (strlen(WayPointDir) == 0) {
        cout << endl << "No waypoint directory found!" << endl;
        return 0;
}

// Get length of WaypointDir
Length = strlen(WayPointDir);

// Saves the folder name into WayPointDir
for (i=0; i<=(Length-9);i++)
        WayPointDir[i]=WayPointDir[i+10];

// Add waypoint directory name to PodDir
strcat(PodDir,"\\");
strcat(PodDir,WayPointDir);

// Change directory to PodBot waypoint directory
SetCurrentDirectory(PodDir);

i = 0;

// Creates a handle and get the first file name
hFind = FindFirstFile("*.pwf", &FindData);

if (hFind == INVALID_HANDLE_VALUE) {
        cout << endl << "Invalid File Handle. No waypoints found!" << endl;
        return 0;
}
```

```
// Saves the first waypoint name
strcpy(WayPointsFound[i],FindData.cFileName);
i++;

do {
        // Get the next file name
        Length = FindNextFile(hFind, &FindData);
        if (Length != 0) {
            strcpy(WayPointsFound[i], FindData.cFileName);
            i++;
        }
    } while (Length != 0); // All waypoints found?


// Close the handle
FindClose(hFind);

// Copy CurDir to PodDir and adds \maps into the end
strcpy(PodDir,CurDir);
strcat(PodDir,"\\maps");

// Change directory to map directory
SetCurrentDirectory(PodDir);

// Creates a handle and get the first file name
hFind = FindFirstFile("*.bsp", &FindData);

if (hFind == INVALID_HANDLE_VALUE) {
    cout << endl << "Invalid File Handle. No maps found!" << endl;
    return 0;
}

// Saves the first waypoint name
strcpy(MapsFound[k],FindData.cFileName);

k++;

do {
        // Get the next file name
        Length = FindNextFile(hFind, &FindData);
        if (Length != 0) {
            strcpy(MapsFound[k], FindData.cFileName);
            k++;
        }
    } while (Length != 0); // All maps found?


// Close the handle
FindClose(hFind);

// Change directory to default
SetCurrentDirectory(CurDir);

// Change prefix om waypointfiles from .pwf to .bsp
for (j1=0; j1<i; j1++) {
    Length = strlen(WayPointsFound[j1]);
    WayPointsFound[j1][Length-3] = 'b';
    WayPointsFound[j1][Length-2] = 's';
    WayPointsFound[j1][Length-1] = 'p';
}
```

```
	// Save resluts in Result
	for (j1=0; j1<i; j1++)
		for (j2=0; j2<k; j2++)
		  if (strcmp(WayPointsFound[j1],MapsFound[j2]) == 0) {
			strcpy(Result[j],MapsFound[j2]);
			j++;
		  }

	// No waypoints for excisting maps, print error
	if (j == 0) {
		cout << endl << "No map has a waypoint file!" << endl;
		return 0;
	}

	// Change directory to default
	SetCurrentDirectory(CurDir);

	for (j1=0; j1 < j; j1++) {
		Length=strlen(Result[j1]);
		Result[j1][Length-4]='\0';
	}

	strcpy(SaveFileName,"mapcycle");
	strcat(SaveFileName,pOrp2);

	// Save the results into mapcycle.pod or mapcycle.txt
	ofstream fout (SaveFileName);
	for (j1=0; j1 < j; j1++)
		fout << Result[j1] << endl;
	fout.close();

	return 1;
}

int NumOfMaps(char *Arg) {
	int PorA,j,k,NumOfMaps=0,NumOfLines=1;
	char ReadBuffer[500][50];

	// Are -a3 or -p3 called?
	if (strncmp(Arg,"-a3",3) == 0)
		PorA = 1;
	else
		PorA = 2;

	// How big number?
	k = strlen (Arg)-4;

	for (j = 0; j<k; j++)
		NumOfMaps += ((int)Arg[4+j]-48)*(int)pow(10,(k-j-1));

	if (NumOfMaps < 1 || NumOfMaps > 500) {
		cout << endl << "The number is lower then 1 or greater then 500!" << endl;
		return 0;
	}

	// Generate a new mapcycle.txt file
	if (PorA == 1) {
		k = All(".txt");
```

```
            if (k == 0)
                return k;
    }
    else {
            k = PodBot(".txt");
            if (k == 0)
                return k;
    }

    // Call the Random function
    Random();

    // Open file
    ifstream fin;

    fin.open("mapcycle.txt");

    // Reads one line at the time from mapcycle.txt
    while (fin.getline(ReadBuffer[NumOfLines-1],50) && NumOfLines != NumOfMaps)
            NumOfLines++;

    // Close file
    fin.close();

    // Save the results into mapcycle.txt
    ofstream fout ("mapcycle.txt");
    for (j=0; j < NumOfLines; j++)
        fout << ReadBuffer[j] << endl;
    fout.close();

    return 1;
}
```