TAWS3

TAWS3 is, as you may have guessed, the 3rd game in a series of "Text Adventures Without Swords". TAWS3 is a strategic fighting game that changes dynamic the more players join in.

Each player has 4 stats: health, strength, stamina, and armor. At the start of the contest each player's stats are equal. Each turn is two phase. In the first phase you can adjust your own stats using one of the following commands:

"status" : see how you are doing.

"players" : see who is playing this contest.

"heal" : Restore lost health up to 2 times your stamina.

"strength" : Increase your attack strength.

"stamina" : Improve your stamina.

"armor" : Improve your armor rating.

"help" : get a hint in game.

Then in phase 2 you choose which player you will attack.

Like chess, this is a somewhat intellectual game, a simple game with depth and complexity, an open strategy game where individual games can last seemly stable for several rounds and fall apart quickly with a clever move. The addition of more players changes the strategy and dynamic dramatically.

TAWS3 was written by "Entar".

/* game.c listing begins: */

```
#include <sys/stat.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

typedef struct
{
  int active;
  int health;
  float stamina, strength, armor;
  char name[64];
} player_t;

player_t *players;
int num_players=0;
int turn=0;
int turn_part=1;
float heal_num;

#define  DEF_STAMINA  10.0f
#define  DEF_STRENGTH  4.0f
#define  DEF_ARMOR  1.0f
#define DEF_HEALNUM  0.5f

int Game_GetPlayer (char *name)
{
  int i;
  for (i=0; i < num_players; i++)
  {
    if (!strcmp(players[i].name, name))
      return i;
  }
  return -1;
}

extern void ZeroReturn (char *input);

void Game_New (void)
{
  char buf[64];
  int i;
  float str=DEF_STRENGTH, stam=DEF_STAMINA, arm=DEF_ARMOR;
  heal_num = DEF_HEALNUM;
  printf("Beginning new game.\n");

  num_players = 0;
  while (num_players < 2)
  {
    printf("Enter the number of players (at least 2): ");
    fgets(buf, 63, stdin);
    ZeroReturn(buf);
    num_players = atoi(buf);
    players = malloc(num_players*sizeof(player_t));
  }
  for (i=0; i < num_players; i++)
  {
    printf("Enter player %i name: ", i+1);
    fgets(players[i].name, 63, stdin);
    ZeroReturn(players[i].name);
  }
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
  printf("Would you like to set custom settings? ");
  fgets(buf, 63, stdin);
  ZeroReturn(buf);
  if (buf[0] == 'y' || buf[0] == 'Y')
  {
    printf("Enter the beginning stamina (Default: %.1f): ", DEF_STAMINA);
    fgets(buf, 63, stdin);
    ZeroReturn(buf);
    if (!buf[0])
      stam = DEF_STAMINA;
    else
    {
      stam = atof(buf);
      if (stam <= 0)
        stam = DEF_STAMINA;
    }

    printf("Enter the beginning strength (Default: %.1f): ", DEF_STRENGTH);
    fgets(buf, 63, stdin);
    ZeroReturn(buf);
    if (!buf[0])
      str = DEF_STRENGTH;
    else
    {
      str = atof(buf);
      if (str < 0)
        str = DEF_STRENGTH;
    }

    printf("Enter the beginning armor (Default: %.1f): ", DEF_ARMOR);
    fgets(buf, 63, stdin);
    ZeroReturn(buf);
    if (!buf[0])
      arm = DEF_ARMOR;
    else
    {
      arm = atof(buf);
      if (arm < 0)
        arm = DEF_ARMOR;
    }

    printf("Enter portion of stamina for healing (Default: %.1f): "
      , DEF_HEALNUM);
    fgets(buf, 63, stdin);
    ZeroReturn(buf);
    if (!buf[0])
      heal_num = DEF_HEALNUM;
    else
      heal_num = atof(buf);
  }
  for (i=0; i < num_players; i++)
    players[i].stamina = stam;
  printf("Beginning stamina set to %.1f\n", stam);

  for (i=0; i < num_players; i++)
    players[i].strength = str;
  printf("Beginning strength set to %.1f\n", str);

  for (i=0; i < num_players; i++)
    players[i].armor = arm;
  printf("Beginning armor set to %.1f\n", arm);

  for (i=0; i < num_players; i++)
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
  {
    players[i].health = players[i].stamina*2;
    players[i].active = 1;
  }

  printf("Portion of stamina for healing set to %.1f\n", heal_num);

  turn = rand()%num_players;
  printf("By random selection, %s goes first.\n", players[turn].name);
  turn_part = 1;
}

int Game_Load (void)
{
  FILE *f;
  char buf[64];
  char filename[96];
  printf("Enter saved game name please: ");
  fgets(buf, 63, stdin);
  ZeroReturn(buf);
  sprintf(filename, "GameData/%s.sav", buf);
  f = fopen(filename, "rb");
  if (f)
  {
    fread(&num_players, sizeof(int), 1, f);
    if (players)
      free(players);
    players = malloc(sizeof(player_t)*num_players);
    fread(players, sizeof(player_t), num_players, f);
    fread(&turn, sizeof(int), 1, f);
    fread(&turn_part, sizeof(int), 1, f);
    fread(&heal_num, sizeof(float), 1, f);
    fclose(f);
    printf("Saved game loaded successfully.\n");
    return 1;
  }
  printf("File not found.\n");
  return 0;
}

void Game_Save (void)
{
  FILE *f;
  char buf[64];
  char filename[96];
  printf("Enter name for saved game please: ");
  fgets(buf, 63, stdin);
  ZeroReturn(buf);
  sprintf(filename, "GameData/%s.sav", buf);
  f = fopen(filename, "wb");
  if (!f)
  {
#ifdef WIN32
    if (mkdir("GameData") == 0)
#else
    if (mkdir("GameData", 0777) == 0)
#endif
    {
      f = fopen(filename, "wb");
      if (!f)
      {
        printf("Save failed.\n");
        return;
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
      }
    }
    else
    {
      printf("Save failed.\n");
      return;
    }
  }
  if (f)
  {
    fwrite(&num_players, sizeof(int), 1, f);
    fwrite(players, sizeof(player_t), num_players, f);
    fwrite(&turn, sizeof(int), 1, f);
    fwrite(&turn_part, sizeof(int), 1, f);
    fwrite(&heal_num, sizeof(float), 1, f);
    fclose(f);
  }
}

int Game_Loop (void)
{
  int i, num;
  char buf[64];
  while (1)
  {
    printf("\n");
    printf ("%s's turn (part %i): ", players[turn].name, turn_part);
    fgets(buf, 63, stdin);
    ZeroReturn(buf);
    if (!strcmp(buf, "help"))
    {
printf("The object of TAWS volume 3 is to eliminate the other players while\n");
printf("keeping yourself alive.  Gameplay is based on the players'\n");
printf("attributes. Damage done is the attacker's strength minus the\n");
printf("defender's armor. A player's maximum health is twice his stamana\n");
printf("-- Gameplay --\n");
printf("There are 2 parts in a turn.\n");
printf("In the first part, the player can choose to heal himself, or\n");
printf("to increment his armor, strength or stamina.\n");
printf("In the second part, the player chooses which player he wishes to\n");
printf("attack.\n");
printf("Part 1 Commands:\n");
printf("heal: heal yourself for %.2f of your stamina.\n", heal_num);
printf("str/strength: increment your strength attribute.\n");
printf("sta/stamina: increment your stamina attribute.\n");
printf("arm/armor: increment your armor attribute.\n");
printf("Part 2 Command:\n");
printf("Type the name of the player you would like to attack.\n");
printf("-- Tips --\n");
printf("You can save or load at any time with the save and load commands.\n");
    }
    else if (!strcmp(buf, "players"))
    {
      printf("%i Players:", num_players);
      for (i=0; i < num_players; i++)
      {
        printf(" %s", players[i].name);
      }
    }
    else if (!strcmp(buf, "status"))
    {
      printf("Which player would you like to see? ");
      fgets(buf, 63, stdin);
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
      ZeroReturn(buf);
      i = Game_GetPlayer(buf);
      if (i != -1)
      {
        printf("%s's status:\n", players[i].name);
        printf("Health: %i/%i\n", players[i].health,
          (int)(players[i].stamina*2));
        printf("Strength: %.1f\n", players[i].strength);
        printf("Stamina: %.1f\n", players[i].stamina);
        printf("Armor: %.1f\n", players[i].armor);
      }
      else
        printf("Player %s not found.\n", buf);
    }
    else if (!strcmp(buf, "save"))
      Game_Save();
    else if (!strcmp(buf, "load"))
      Game_Load();
    else if (!strcmp(buf, "exit") || !strcmp(buf, "quit"))
      break;
    else if (turn_part == 1) // heal or increment a stat
    {
      if (!strcmp(buf, "heal"))
      {
        players[turn].health += (int)(players[turn].stamina*heal_num+0.5f);
        if (players[turn].health > (int)(players[turn].stamina*2))
          players[turn].health = (int)(players[turn].stamina*2);
        turn_part = 2;
        printf("%s's health increased to %i", players[turn].name,
          players[turn].health);
      }
      else if (!strcmp(buf, "str") || !strcmp(buf, "strength"))
      {
        players[turn].strength += 1.0f;
        turn_part = 2;
        printf("%s's strength increased to %.1f", players[turn].name,
          players[turn].strength);
      }
      else if (!strcmp(buf, "sta") || !strcmp(buf, "stamina"))
      {
        players[turn].stamina += 1.0f;
        players[turn].health += 2;
        turn_part = 2;
        printf("%s's stamina increased to %.1f", players[turn].name,
          players[turn].stamina);
      }
      else if (!strcmp(buf, "arm") || !strcmp(buf, "armor"))
      {
        players[turn].armor += 1.0f;
        turn_part = 2;
        printf("%s's armor increased to %.1f", players[turn].name,
          players[turn].armor);
      }
    }
    else if (turn_part == 2) // attack
    {
      i = Game_GetPlayer(buf);
      if (i == turn)
      {
        printf("You can't attack yourself, stupid!");
        i = -1;
      }
      else if (i == -1)
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
      {
        printf("Player %s not found.  Try again.\n", buf);
      }
      else // hit someone for reals
      {
        num = (int)(players[turn].strength-players[i].armor);
        printf("%s damaged %s for %i damage!\n", players[turn].name,
          players[i].name, num);
        players[i].health -= num;
        if (players[i].health < 1)
        {
          printf("%s has defeated %s!\n", players[turn].name, players[i].name);
          players[i].active = 0;
        }
        num = turn;
        turn_part = 1;
        do
        {
          turn++;
          if (turn >= num_players) // loop back
            turn = 0;
        }
        while (players[turn].active == 0);

        // cycled through, back where we started, he's last man standing
        if (turn == num)
        {
          printf("%s is victorious!\n", players[turn].name);
          break;
        }
      }
    }
  }

  printf("\n");

  // game has come to a conclusion, but don't exit yet
  if (players)
    free(players);
  return 1;
}
```

/* main.c listing begins */

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

extern void Game_New(void);
extern int Game_Load(void), Game_Loop(void);

void ZeroReturn (char *input) {
  char *c = strchr(input, '\n');
  if (c)
    *c = 0;
}

int main (int args, char *argc[]) {
  char buf[32];
  printf("\n");
  printf("**********************************************************\n");
  printf("Welcome to TAWS (Text Adventure Without Swords) volume 3!\n");
```

/* Listing continued on next page…*/

/* Listing continued from previous page */

```
  printf("************************************************************\n");
  printf("\n");

  srand(time(NULL));

  while (1) {
    while (1) {
      printf ("Commands:\n");
      printf ("new: Begin a new game.\n");
      printf ("load: Load a saved game.\n");
      Printf ("exit: Exit TAWS v3, but why would you want that?\n");
      printf ("Please enter a command: ");
      fgets(buf, 31, stdin);
      ZeroReturn(buf);
      if (!strcmp(buf, "new") || !strcmp(buf, "New")) {
        Game_New();
        break;
      } else if (!strcmp(buf, "load") || !strcmp(buf, "Load")) {
        if (!Game_Load())
          continue;
        else
          break;
      } else if (!strcmp(buf, "exit") || !strcmp(buf, "quit")) {
        printf ("Well wasn't that fun? See you next time!\n");
        return 0;
      } else if (!strcmp(buf, "TAWS")) {
        printf("TAWS is most commonly known as Text Adventure Without
Swords.\n");
        printf("However, it has other meanings as well, including:\n");
        printf("Terribly Amazing Walrus Strategy\n");
        printf("Torrential Admission of Watery Soup\n");
        printf("Timetable Antithesis Waking Somebody\n");
        printf("Top Ankle Wise Surrender\n");
        printf("... or just TAWS.\n");
      }
    }

    if (Game_Loop() == 0)
      break;
  }

  return 0;
}
```