Scorched Everitt

This Site is dedicated towards our ECE291 Final Project: Scorched Everitt, which is based off the classic DOS game "Scorched Earth," programmed by Wendell Hicken in the early 1990's. However, our game will have a little bit of an ECE twist to it. Will time allotted, we may implement serial or TCP/IP networkability for multiplayer games. No code or material whatsoever will be taken from the original Scorch game. To find more information about the inspiration of our game, please visit http://www.classicgaming.com/scorch/.

..::| Team Members |::..

Suneil Hosmane: Game Engine, Physics, I/O Control

Terrence Janas: Graphics, Multimedia Design, Webmaster

Yajur Parikh: Physics, Intro

Problem Description

Key issues to work out:

* Mapping the X,Y coordinates of the projectile to the right pixel size on the screen.
  i.e., making the projectile scaled accurately
* Implementing the sound routine with the game
* Making all the functions jive with each other
* Complete a WORKING project in 3 weeks!!

Implementation

The game begins with a small intro, followed by the main menu. The player is

allowed four different options... PLAY GAME, OPTIONS, CREDITS, and EXIT GAME. If the player decides OPTIONS, they will be directed towards another menu where they can either turn on and off sound, turn on and off wind, choose how many rounds they want to play, etc.. All option parameters will be saved so that when the game is over, and the player decides to play another game, the same options will apply. CREDITS just show who did what in the game. EXIT GAME exits to DOS. PLAY GAME starts game play. First player 1 chooses his/her tank, then associates a name to this tank. Likewise, player 2 does the same. Then the game starts!! Each player keeps firing till either they get killed by their own bullet, or by the other player. You score will depend on the following formula, (1600 - failed attempts to kill). If you score is 1500, 3000, 4500, etc... your gun size will increase (meaning your blast radius will increase). The game will notify you of level up. At the end of each round, a winner screen will pop up indicating who won, and how many points they will receive this round. At the end of the game, another screen will pop up show who won, point totals, and games won. Also, at the end of the game the player will be able to choose to quit the game, return to the main menu, or player another set of rounds.

STRUCTURE OF THE GAME

```
_GameStart()

void _Intro()
void _InitGameVariables

.MainMenu

    void _MainMenu()
    cmp byte [_MenuItem], PLAY_GAME
    je near .Play

    cmp byte [_MenuItem], OPTION_MENU
    je near .Option

    cmp byte [_MenuItem], CREDIT
    je near .Credit

    cmp byte [_MenuItem], EXIT
    je near .MainDone

.Play:

     void _Player1Setup()
     void _Player2Setup()
     void _InitializeGame()
```

```
    void _PlayGame()

   cmp byte [_Exit], 0
   je near .MainMenu

   jmp near .MainDone

.Option:
   void _OptionsMenu
   jmp near .MainMenu

.Credit:
   void _Credit()
   jmp near .MainMenu

.MainDone:

   ret
```

PROTO-TYPE FUNCTIONS

void _Scorch()

Inputs:	none
Function:	This is the very basic shell of the game. It invokes the menus, starts the game, and handles the exit cases.
Author:	Suneil Hosmane

void _InitGameVariables()

Inputs:	none
Function:	Initialize all the option menu flags to their initial values. Sound = OFF, Wind = OFF, Rounds = 5, Color = Default
Author:	Suneil Hosmane

void _OptionsMenu()

Inputs:	none
Function:	Let the user be able to: (1) Turn ON, and OFF the following: SOUND, WIND (2) Pick between 1, 3, 5, 7, or 9 Rounds of Combat (3) Pick between different sky background colors
Author:	Suneil Hosmane

void _InitializeGame(dword *Screen, dword *Tank1, dword *Tank2)

Inputs:	Screen is offset to some buffer P1TankOffset is Offset to Player 1's Tank P2TankOffset is Offset to Player 2's Tank
Function:	Draws the main game screen, with land, background color, and tanks

Author: Terrence Janas

void _SetScoreGun(word Player, word Turns)

Inputs: Player = Player 1 or Player 2
Turns = How many turns it took to hit someone

Function: Depending on how many turns the player took, we obtain a different round score. The formula is 1600 - # turns. In addition to calculating the round score, it also displays on the screen LEVEL UP if the player's cumulative score 1500, 300, etc...

Author: Yajur Parikh

void _FillGunBox(word Player, word Score)

Inputs: Player = Player 1 or Player 2
Score = The end of round score

Function: Depending on the player's score, we determine by how much (or if any) we increase the players gun detonation size. Also, the end of round score is added on to the right player's total score.

Author: Terrence Janas

void _PlayGame()

Inputs: none

Function: This is the main function essentially. It directs all traffic during the game. Sets up a game, and then calls all the end of round/end of game screens.

Author: Suneil Hosmane

void _DrawProjectile()

Inputs: none

Function: When a person has hit the enter button we can draw the projectile using the physics equations we learning in Physics 111, with modification of course. We can also determine whether or not a player has been hit

Author: Suneil Hosmane & Yajur Parikh

void _DrawBullet(dword *DestOff, word DestWidth, word DestHeight, word X1, word Y1, word X2, word Y2, dword Color)

Inputs: X1 & X2 Coords, Y1 & Y2 Coods, Destination Offset, Destination Height, Destination Width, and Color

Function: Similar to DrawLine(), except for the fact that it stops

drawing once you have hit terrain or another player.

Author: Yajur Parikh

void _Player1Set( )

Inputs: none

Function: Handles moving around the tank, setting the power & angle, and displays the help & premature quit menus if need be.

Author: Suneil Hosmane

void _Player2Set( )

Inputs: none

Function: Same as Player1Set(), except is for player 2

Author: Suneil Hosmane

dword _AllocateMemory( )

Inputs: none

Outputs: eax contains 0 on success, -1 on error

Function: Allocate memory for all the variables that need memory allocated.

Author: Yajur Parikh

void _MainMenu(dword *DestOff, word Width, word Height)

Inputs: Destination to a screen buffer, width of buffer, height of buffer

Function: Draws the main, and depending on what was picked, sets _MenuItem to the right value; 0 - Play Game, 1 - Options, 2 - Credits, 3 - Exit Game

Author: Terrence Janas

void _DrawMainScreen(dword *DestOff)

Inputs: Destination to a screen buffer

Function: Draws the crude game screen and outputs to the buffer/

Author: Suneil Hosmane

void _Player1Setup()

Inputs: none

Function: player 1 selects his/her tank, and then enters a name

Author: Suneil Hosmane

void _Player2Setup()

Inputs: none

Function: same as player1setup(), except for player 2

Author: Suneil Hosmane

void _GetLand(a lot of stuff :-))

Inputs:
GameWindow = Offset to the mini window in the screen
GameW = GameWindow Width
GameH = GameWindow Height
Screen = Offset to the screen
ScreenW = Screen Width
ScreenH = Screen Height
Num = The # of the land (0-9)
SkyColor = Color of the sky

Function: Takes an integer (Num), and copies that land # from the land.png to the buffer specified

Author: Suneil Hosmane

void _DrawString(dword *StringOff, dword *Screen, word X, word Y, dword Color)

Inputs:
StringOff = String
Screen = some buffer
X = X coordinate
Y =Y coordinate
Color = Color of Text

Function: Displays a string on the screen (uses DrawText), except it draws the characters one block at a time (reduces flickering on the screen)

special characters:
Semi-Colon - function skips this character

Author: Yajur Parikh

void _DrawTank(dword *TankOff, word TankPos, dword *Screen, word LandNumber)

Inputs:
TankOff = Offset to buffer with tank images
TankPos = Tank Positions (0-9)
Player = Player#
Screen = screen buffer
LandNumber - Which terrain are we on?

Function: Displays the battle tank in the right place, and right position.

Author: Terrence Janas

void _GetTankBK(word Player, dword *Screen, word LandNumber)

Inputs: Player = Player 1 or Player 2
Screen = screen buffer
LandNumber - Which terrain are we on?

Function: Obtains the 60x60 square chunk of the block, so that when we
redraw the tanks, you only redraw the 60x60 slice to reduce flickering

Author: Terrence Janas

void _DisplayScore(word Score, dword *Screen, word X, word Y)

Inputs: Score = Player's Score
Screen = screen buffer
X, Y = (x,y) coordinates to display score

Function: Writes player's score to the buffer specified

Author: Suneil Hosmane

void _DisplayPowerAngle(word FLAG, word PowerOrAngle, dword *Screen, word X, word Y)

Inputs: Flag = Look at Function
PowerOrAngle = Either the power or angle
Screen = screen buffer
X, Y = (x,y) coordinates to display either power/angle

Function: Flag = 0; Display all zeros
Flag = 1; Increment PowerOrAngle by 1
Flag = 2; Decrement PowerOrAngle by 1
Flag = 3; JUST DISPLAY

write either an angle or power to the buffer.

Author: Yajur Parikh

void _ReturnAsciiChar()

Inputs: al holds an integer

Function: converts al to its ascii equivalent, and returns it back into al

Author: Terrence Janas

void _Delay(dword NumTicks)

Inputs: NumTicks = Number of Ticks

Function: Delays the program by the number of ticks specified

Author: Edwin Daniels (previous ECE291 student)

word _InstallTimer()

Outputs: eax = 1 if error, 0 otherwise

Function: Installs TimerISR

Author: Terrence Janas

void _RemoveTimer()

Inputs: none

Function: Uninstalls TimerISR and restores original handler

Author: Terrence Janas

void _TimerISR(dword TimerTicks)

Inputs: TimerTicks = Number of Ticks

Function: Handles timer ticks from the system timer

Author: Edwin Daniels (previous ECE291 student)

void _LevelUpSound()

Inputs: none

Function: Play a very short melody when a player levels up.

Author: Terrence Janas

void _Random(word MaxNum)

Inputs: MaxNum = Maximum Random Rumber

Function: returns a randomly-generated number within bounds specified by MaxNum

Author: Given to ECE 291 Spring 2001 Semester's class

void _GetWind(word Wind)

Inputs: Wind

Function: Randomly generates a wind, and passes it back in Wind

Author: Terrence Janas

void _Firing Sound()

Inputs: none

Function: Play a very short melody when a player fires weapon.

Author: Terrence Janas

void _ImpactSound()

Inputs: none

Function: Play a very short melody when a player gets hit

Author: Terrence Janas

void _BattleHymn()

Inputs: none

Function: Play the first 7 measures of the Battle Hymn of the Republic

Author: Terrence Janas

void _Intro()

Inputs: none

Function: Intro scene where tank moves across screen & fades.

Author: Terrence Janas

We also used every function that we had to write for MP4.
To view these functions and their descriptions,
please visit http://courses.ece.uiuc.edu/ece291/mp/mp4/
