Simple ЯTS GΔme Dεsign & Cөding

OpenEdition

June 6th, 2003

Chapter I: Introduction

Why did I write this book ?

Trong thời điểm hiện tại, có rất nhiều bạn trẻ hướng sự quan tâm của mình vào thiết kế và lập trình game, phải công nhận một điều rằng chơi game đã là một sự hấp dẫn, nhưng chơi game mình làm ra còn hấp dẫn hơn nhiều, cho dù đó là một trò chơi dở ẹc. Trên những diễn đàn tin học trên Internet đầy rẫy những topic kiểu như là "Lập trình game với.." hay là "Ai chỉ giúp em lập trình game như thế nào", tất cả đều là sự quan tâm đến mục đích "Làm thế nào để viết nên một game ?". Những ý kiến xoay quanh vấn đề này rất đa dạng, nói khó cũng có mà nói…dễ cũng có. Với kinh nghiệm và trình độ hiện tại, tôi công nhận một điều là khó, nhưng không phải là quá khó nếu bạn biết đặt mục tiêu vừa tầm và tìm cách tiếp cận nó. Nếu ai cũng hi vọng mình viết ra game tương đương với Quake hay là Warcraft – là sản phẩm của cả một tập thể chuyên nghiệp được xây dựng trong một thời gian dài - ngay lập tức thì quá là viển vông, nhưng nếu bạn hạ thấp mục tiêu xuống một chút như là làm những game đơn giản kiểu như bắn tăng hay Mario, IQ hoặc là những game cho trẻ em trước thì không quá khó nếu bạn biết lập trình. Cá nhân tôi dường như đã nhận thức được điều này và đã có được một số thành công nhỏ trong việc thiết kế và lập trình game. Trong thời điểm này tôi đang được nghỉ ôn thi và tôi quyết định dành một thời gian ngắn để viết cuốn sách này, trong cuốn sách này tôi trình bày một cách đầy đủ quá trình tôi thiết kế và viết mã cho một game RTS đơn giản (được lược bỏ khá nhiều tính năng ví dụ như chơi mạng) nhằm giúp các bạn hiểu được phần nào quá trình "ziết game tại gia". Hi vọng cuốn sách có thể giúp các bạn trong quá trình thiết kế và viết game của chính mình. Chúc tất cả các bạn thành công trong dự án viết game của mình.

What is RTS game ?

RTS – Realtime Strategy - Thể loại game chiến thuật thời gian thực mà khởi đầu với dòng game Dune, Command & Conquer của Westwood Studio và đã nhanh chóng trở thành dòng

game ăn khách trên toàn thế giới và cả tại Việt Nam. Thời điểm hiện tại sự phát triển của dòng trò chơi này dường như chững lại, do không có nhiều tiến bộ vượt bậc như thời điểm Starcraft (Blizzard Entertainment) ra đời mà chỉ xuất hiện những game lai tạp với các thể loại khác (Warcraft III là thể loại Roleplay Strategy) nên dường như dòng game này đang dần nhường bước cho các dòng game ăn khách khác. Nhưng cá nhân tôi vẫn ôm ước vọng có thể xây dựng một game RTS ăn vào bối cảnh lịch sử Việt Nam và đem lại cho người chơi nhiều cải tiến quan trọng trong cách chơi (He he, cải tiến quan trọng à nha, nhưng không nói đâu, lúc nào làm xong sẽ…nói, hì hì).

How this book will help you

Cuốn sách này tôi viết lại quá trình thiết kế, viết mã – dĩ nhiên là chú thích tương đối cụ thể – cho một game RTS được thiết kế đơn giản tối đa. Bên cạnh đó tôi cũng đưa ra một số kinh nghiệm cá nhân có được trong quá trình làm việc nhằm giúp các bạn có thể hiểu được phần nào công việc thiết kế và lập trình game (Không chuyên nghiệp). Còn cuốn sách này giúp gì được bạn thì còn tuỳ thuộc vào bạn, nếu mà bạn thuộc hàng cao thủ bít hết rùi hoặc là bạn dumb quá đọc chẳng hiểu gì thì dĩ nhiên là nó chẳng giúp gì được bạn đâu. Tôi sẽ cố gắng giải thích cặn kẽ nhất nhằm giúp các bạn có thể hiểu được chương trình làm việc như thế nào. Tất cả những gì còn lại đều phụ thuộc vào bạn.

Readers requirements

Trong cuốn sách này, tôi sử dụng ngôn ngữ Delphi để viết chương trình. Nhưng thực tế hoàn toàn lập trình không hề sử dụng đến các component và cũng không hề sử dụng đến Window API mà thông qua thư viện xây dựng riêng nên gần như các bạn chỉ cần biết Object Pascal là có thể hiểu được mã của chương trình làm việc như thế nào, tôi cố gắng viết mã một cách chân phương nhất có thể – không hề tối ưu, tôi có ít thời gian wé - nhằm mục đích để mọi người đều có thể cảm thấy dễ hiểu và có thể chuyển sang ngôn ngữ khác, theo tính toán của tôi thì chương trình này có thể chuyển sang mọi ngôn ngữ lập trình bậc cao thông dụng như là VB, C, C++, C# hoặc Java. Về mặt đồ hoạ tôi không sử dụng những hệ thống 3D phức tạp mà chỉ thiết kế với hệ thống 2D và sprite animation đơn giản, chắc chắn đa số các bạn có thể hiểu được kĩ thuật thô sơ này. Mã nguồn và dữ liệu của game được cung cấp kèm theo đầy đủ, các bạn chỉ cần biên dịch lại là có thể chạy được ngay lập tức.

System & Software requirements

- Hệ thống PIII 500Mhz, 64MB RAM, card màn hình 8MB, màn hình hỗ trợ chế độ phân giải 800x600x16bit màu, có card sound (Thực tế tôi chưa chạy thử trên máy có cấu hình yếu hơn nên không rõ có chạy được hay không).
- Windows 95, 98, 98SE, NT, 2000 hoặc XP.
- DirectX 8.1 trở lên.
- Biên dịch bằng Borland Delphi 6 hoặc Borland Delphi 7.
- Nếu các bạn cần chỉnh sửa dữ liệu của game thì cần thêm một trình soạn thảo ảnh (Photoshop, ACD FotoCanvas…).

Misc

- Tôi viết cuốn sách này chủ yếu hướng đến các bạn gần như chưa biết gì về lập trình game và chỉ có dụng ý chia sẻ chút kiến thức ít ỏi của bản thân, nếu trong quá trình biên tập có chút sơ sót mong các bạn lượng thứ, nếu như cuốn sách này làm bạn cảm thấy hứng thú hoặc muốn trao đổi thêm với tôi các bạn có thể liên lạc qua địa chỉ e-mail kimngan2508@yahoo.com (Đây là địa chỉ mirror thôi, đừng bomb tui nha, he he) hoặc qua YIM: kimngan2508.

- Cuốn sách này tôi viết hoàn toàn không nhằm mục đích thương mại, toàn bộ mã nguồn chương trình và cuốn sách này được phân phối miễn phí, tôi không chấp nhận mọi hành vi kinh doanh dựa trên tập sách và chương trình này. Đối với chương trình, các bạn có thể tuỳ nghi sử dụng, nâng cấp, chỉ cần để một record ghi chú là nâng cấp từ chương trình của CrazyBabe là được. Nhưng tôi nghĩ các bạn nên viết lại hoàn toàn thì hơn (nếu muốn làm game thực sự) vì chương trình này không được tốt.
- Trong chương trình, vì ngại thiết kế hình ảnh và âm thanh nên tôi mượn tạm hình ảnh và âm thanh trong loạt game Starcraft của nhà sản xuất Blizzard Entertainment (Hì hì, chắc là chả ai bít nên sẽ không bị..kiện).

Chapter II: Game Design

Design a very simple RTS game

Như đã nói trước đây, tôi viết cuốn sách này trong một khoảng thời gian ngắn (Chỉ bốn ngày cả viết mã và viết tài liệu) nên đối tượng game tôi sử dụng phải tương đối đơn giản (Do lí do hạn chế thời gian và cả trình độ nữa, hì hì…). Ở đây tôi thiết kế một game thể loại RTS 2D nhưng tính năng rất hạn chế. Tôi xây dựng game này hiện tại hoàn toàn không hỗ trợ chế độ multiplayer, nhưng hi vọng trong thời gian đến tôi sẽ có thời gian để nâng cấp game lên hỗ trợ chế độ này. Và tôi cũng loại bỏ một số yếu tố mà tôi cho là đơn giản như là chế độ menu, chọn bản đồ, loại quân… mà nhảy vào là chương trình load một bản đồ mặc định và chạy luôn bản đồ này cho đến khi trận chiến kết thúc hoặc là bạn.. chán wá thoát ra ngoài – he he - tức là tôi chỉ quan tâm duy nhất đến chế độ battle của trò chơi mà thôi, bỏ qua tất cả các yếu tố khác.

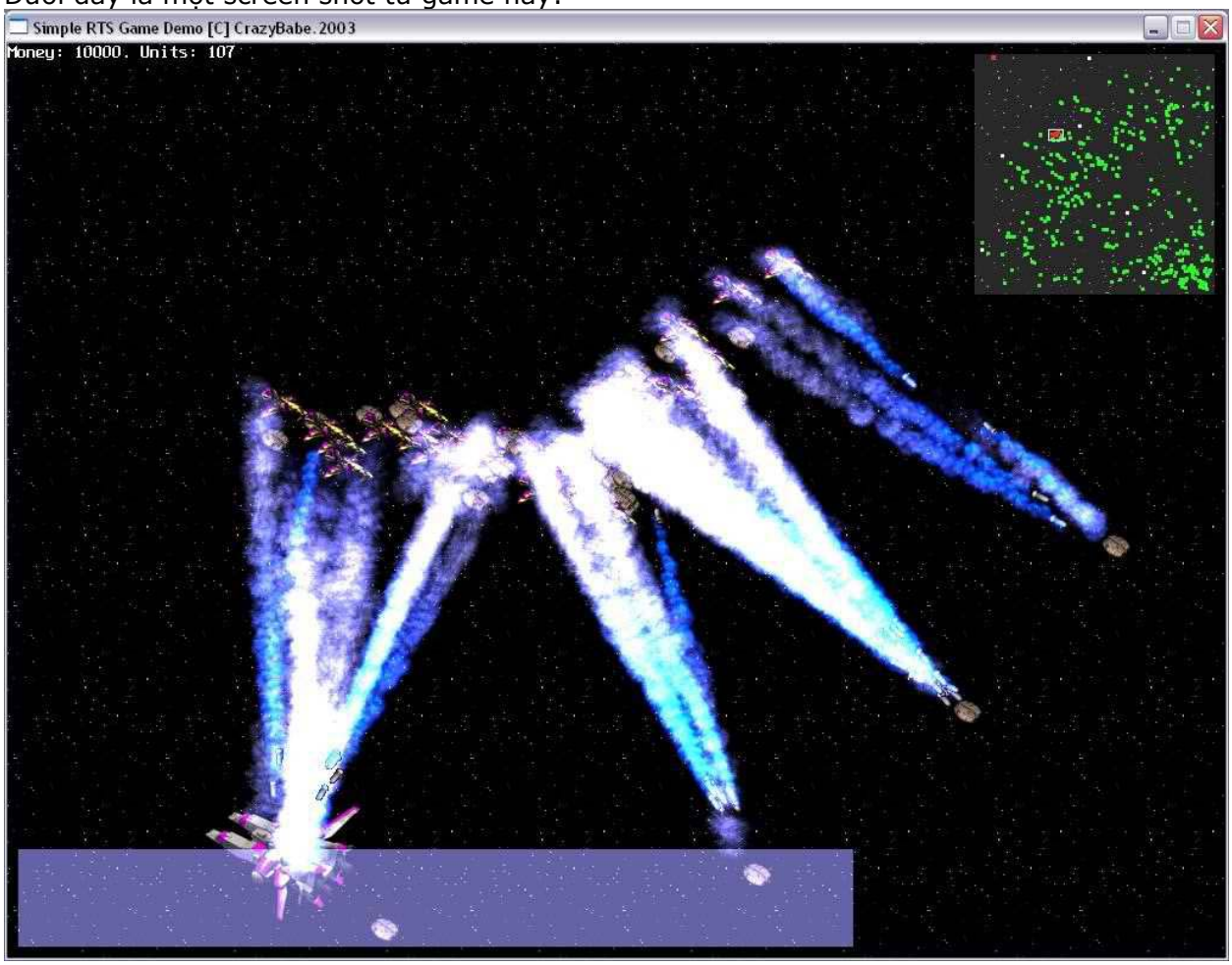
Một game RTS cơ bản thì xoay quanh một số yếu tố như sau: cơ chế khai thác tài nguyên, cơ chế sinh sản các đơn vị quân, tính năng của các đơn vị quân và cách chúng tấn công, kiểu địa hình cũng là một yếu tố quan trọng.

Game này tôi thiết kế sơ bộ như sau:

- Thể loại chiến tranh trong không gian - Starwars lun, chiến chưa ? :)
- Không có địa hình, loại bỏ bớt yếu tố phức tạp cho địa hình. Cái này cũng nhằm để dễ lập trình nữa.
- Game gồm hai loại đơn vị chính là tàu mẹ và tàu con. Tàu mẹ là trung tâm đảm nhận việc xây dựng tàu con, lưu giữ tài nguyên… Tàu con thì đảm nhận nhiều nhiệm vụ khác nhau như là tuần tiễu, tấn công, khai thác…
- Các đơn vị tàu có các tính năng như là di chuyển, tấn công, tuần tiễu, khai thác, xây tàu con.
- Hệ thống phím tắt cố định (Tôi bỏ qua việc xây dựng hệ thống menu điều khiển).
- Hệ thống xây dựng game là 2D.

Sơ bộ là như vậy, các tính năng khác tôi nghĩ rằng các bạn có thể thêm vào đơn giản nếu như hiểu được chương trình. Nếu muốn nâng cấp game thành 3D tôi nghĩ cũng có thể làm được.

Dưới đây là một screen shot từ game này:



Game programming environment

Các bạn khi trao đổi với nhau về lập trình game thường trao đổi với nhau về những vấn đề cụ tỉ như là vẽ bằng DirectX hay OpenGL, làm thế nào để lấy một sprite hay là vẽ sprite đấy bằng hiệu ứng như thế nào…thiết nghĩ những thứ đó quá lặt vặt – dĩ nhiên là không nói đến chuyện thiết kế các engine 3D lớn rùi – Trong khi lập trình game này, tôi sử dụng bộ thư viện AvenusHelper được tôi thiết kế nhằm giảm tối đa các công việc các bạn cần thực thi chi tiết với hệ thống như là khởi tạo đồ hoạ, đọc ảnh và vẽ ảnh bằng những hiệu ứng đơn giản như là add, sub, mul…Ở đây các bạn chỉ cần chú ý đến việc làm thế nào game chạy được, các unit được xử lý như thế nào, máy tính chơi ra sao, những sự kiện được xử lý thế nào…mà thôi. Nếu cần quan tâm đến những chi tiết nhưng là làm thế nào để thực hiện hiệu ứng alpha blend chẳng hạn chúng ta sẽ bàn sau - ở một tập sách khác.

Chapter III: Game Coding

Chương này chỉ đơn giản là tôi chú thích mã nguồn của chương trình.

Dưới đây là mã nguồn của đơn vị Game, là thành phần chính của chương trình.

```
{
  Tôi đặt định nghĩa này nhằm chỉ chương trình chạy trong chế độ toàn màn  hình hay chế độ cửa sổ
}
{$Define FullScreen}
{
  Định nghĩa này nhằm xác định chương trình có hỗ trợ âm thanh hay không
}
{$Define SoundPlaying}
UNIT Game;

INTERFACE
{
  Ở đây tôi khai báo các đơn vị mà chương trình sử dụng
}
USES Windows,
     MMSystem,
     SysUtils,
     Math,
     FireStormDLL,
     AvenusHelperDLL;

CONST
{
  ScrWidth x ScrHeight là hằng định nghĩa kích thước cửa sổ hoặc độ phân giải ở chế độ toàn màn hình. Chú ý là khi chương trình ở chế độ cửa sổ tức là biên dịch không có chỉ thị FullScreen thì kích thước là tuỳ ý, nhưng nếu chương trình chạy ở chế độ toàn màn hình thì kích thước độ phân giải này phải được màn hình hỗ trợ và thường là các độ phân giải thông dụng như là: 640x480, 800x600, 1024x768, 1152x864, 1280x1024…
```

```
}
  ScrWidth                    = 1024;
  ScrHeight                   = 768;
{
  Caption là hằng xâu định nghĩa tên cửa sổ của chương trình tạo ra
}
  Caption                     = 'Simple RTS Game Demo [C]CrazyBabe.2003';
{
  MaxUnit là hằng số định nghĩa số lượng unit lớn nhất mà chương trình hỗ trợ xử lý, con số
này là tui đặt đại như zậy chớ nếu mà xử lý thực tế như thế (chương trình hoàn toàn không
tối ưu) thì chỉ cần khoảng 4000 chắc là con P4 của tui cũng die hard lun chứ không cần phải
thử trên máy cấu hình yếu đâu :>
}
  MaxUnit                     = 8000;
{
  DataPath là đường dẫn đến thư mục chứa dữ liệu của chương trình
}
  DataPath                    = 'Data\';
{
  ShipsSettingDir là đường dẫn đến thư mục con chứa các file dữ liệu về các loại tàu trong
game.
}
  ShipsSettingDir             = 'ShipsSetting\';
{
  ScrollSpeed là tốc độ di chuyển minimap trong game, tôi đặt nó là biến số nhằm mục đích
có thể thay đổi tốc độ nhưng phần này tui chưa viết, các bạn tự thêm vào nha.
}
  ScrollSpeed                 : Word = 15;
{
  MoveFarLeng, TooNearLeng, AroundLeng là các hằng số chỉ độ lớn giới hạn ví dụ như là
giới hạn của các tàu quá gần nhau…
}
  MoveFarLeng                 = 10;
  TooNearLeng                 = 3;
  AroundLeng                  = 100;
{
  ScreenUpDate là khoảng thời gian (phần nghìn giây) mà chương trình thực thi cập nhật lại
màn hình. Với giá trị 33 chương trình cập nhật khoảng 28-30 lần một giây.
  UnitUpDate là khoảng thời gian mà chương trình thực hiện xử lý cho các đơn vị, với giá trị
10 thì chương trình xử lý các đơn vị trung bình khoảng 70 lần một giây.
  InputUpDate, MiniMapUpDate lần lượt là khoảng thời gian chương trình thực hiện cập nhật
bàn phím, chuột… và cập nhật lại minimap (Bản đồ con).
  MoneyTime là khoảng thời gian chương trình thêm xiền cho các phe.
}
  ScreenUpdate                : LongWord = 33;
  UnitUpdate                  : LongWord = 10;
  InputUpdate                 : LongWord = 10;
  MiniMapUpdate               : LongWord = 10;
  MoneyTime                   : LongWord = 120000;
{
  DefaultMoney là số xiền game cho mỗi phe mỗi lần "cấp phát".
}
  DefaultMoney                = 10000;
{
  PanelSizeX, PanelSizeY là kích thước Panel thể hiện thông tin về đơn vị đang được chọn.
  MiniMapSizeX, MiniMapSizeY là kích thước bản đồ nhỏ hiển thị trên màn hình.
```

```
}
  PanelSizeX                = 700;
  PanelSizeY                = 80;
  MiniMapSizeX              = 200;
  MiniMapSizeY              = 200;
{
  Cứ sau khoảng FrameRateForFindTarget frame thì chương trình thực hiện tìm kiếm mục tiêu cho các đơn vị đang không có lệnh.
}
  FrameRateForFindTarget    = 40;

TYPE
{
  Định nghĩa kiểu để đánh số các đơn vị trong game.
}
  TUnitCount = 1..MaxUnit;
{
  Định nghĩa kiểu toạ độ nguyên và toạ độ thực.
}
  TRPoint = Record
    X,Y : Single;
  End;
  TPoint = Record
    X,Y : Integer;
  End;
{
  Định nghĩa số hướng đơn vị có thể xoay là 32 hướng. Số hướng này càng nhiều thì đơn vị xoay càng mịn, nếu xây dựng với 3D thì không cần đến định nghĩa số hướng này mà chỉ cần lưu lại góc nhìn của đơn vị. Hic, chóng mẹt wé…
}
  THeading = (
    H01,H02,H03,H04,
    H05,H06,H07,H08,
    H09,H10,H11,H12,
    H13,H14,H15,H16,
    H17,H18,H19,H20,
    H21,H22,H23,H24,
    H25,H26,H27,H28,
    H29,H30,H31,H32);
{
  Định nghĩa số kiểu tàu có trong game, hik, bạn bịa tên gì cũng ok…
}
  TSpaceShip = (
    NoneShip,
    FirePlanet,
    SupperBattleShip,
    BattleCruiseAmon,
    BattleCruiseBaton,
    CooNok,
    Scout,
    Interceptor,
    Carrier,
    MiniMissile,
    EvelMissile,
    MileMissile,
    Burn
  );
{
```

Định nghĩa số lệnh mà unit có thể thực hiện trong game, do game tôi thiết kế rất đơn giản nên unit mới chỉ có thể nhận các lệnh như là di chuyển, đi tuần, tấn công, chuyển dịch (đi đến đích cụ thể nhưng tấn công địch gặp trên đường)…
}
```
  TCommand = (
    NoCmd,
    //
    //Mouse command
    //
    MouseSelection,
    //
    //Unit command
    //
    CmdMove,
    CmdAttack,
    CmdAttackP,
    CmdAttackT,
    CmdAttackM,
    CmdPatrol,
    CmdFire,
    CmdBurn,
    CmdEnd//Temporary command
  );
```
{
Định nghĩa số phe trong game, bốn phe Player1-4 dùng để đánh dấu các phe chơi, còn NonePlayer dùng để đánh dấu những unit misc như là khói, tên lửa… PlayerNature để đánh dấu các vật thể như là các hành tinh, thiên thạch…
}
```
  TPlayer = (
    NonePlayer,
    PlayerNature,
    Player1,
    Player2,
    Player3,
    Player4
  );
```
{
Định nghĩa các kiểu súng mà unit sử dụng
}
```
  TGun = (
    GunMiniMissile,
    GunEvelMissile,
    GunMileMissile
  );
```
{
Định nghĩa các tên kiểu, tên hướng… để lấy giá trị từ file config.
}
```
//Defined a name of types for configuration getting types
CONST
  HeadingName : Array[THeading] of String[5] = (
    '[H01]','[H02]','[H03]','[H04]',
    '[H05]','[H06]','[H07]','[H08]',
    '[H09]','[H10]','[H11]','[H12]',
    '[H13]','[H14]','[H15]','[H16]',
    '[H17]','[H18]','[H19]','[H20]',
    '[H21]','[H22]','[H23]','[H24]',
    '[H25]','[H26]','[H27]','[H28]',
    '[H29]','[H30]','[H31]','[H32]');
  PlayerName : Array[TPlayer] of String[8] = (
    '',//NonePlayer
    'NATURE',
```

```
    'PLAYER01',
    'PLAYER02',
    'PLAYER03',
    'PLAYER04'
  );
  ShipName : Array[TSpaceShip] of String[20] = (
    '',//NoneShip
    'FIREPLANET',
    'SUPPERBATTLESHIP',
    'BATTLECRUISEAMON',
    'BATTLECRUISEBATON',
    'COONOK',
    'SCOUT',
    'INTERCEPTOR',
    'CARRIER',
    'MINIMISSILE',
    'EVELMISSILE',
    'MILEMISSILE',
    'BURN'
  );
  GunName : Array[TGun] of String[20] = (
    'MINIMISSILE',
    'EVELMISSILE',
    'MILEMISSILE'
  );
{
  Định nghĩa một số tên chỉ thị để lấy thông tin từ file config.
}
//Constant name of script directive
CONST
  SettingStr               = 'SETTING';
  ImageFilesStr            = 'IMAGEFILES';
  ShipFrameStr             = 'SHIPFRAME';
  ExplosionSkinStr         = 'EXPLOSIONSKIN';
  ShipGunStr               = 'SHIPGUN';
  ShipDamageStr            = 'SHIPDAMAGE';
  ShipAttackRangeStr       = 'ATTACKRANGE';
  ShipNameStr              = 'SHIPNAME';
  ShipSizeStr              = 'SHIPSIZE';
  ShipSpeedStr             = 'SHIPSPEED';
  ShipWaitStr              = 'SHIPWAIT';
  ShipTurnWaitStr          = 'SHIPTURNWAIT';
  ShipHitPointStr          = 'SHIPHITPOINT';
  ShipCanCreate            = 'SHIPCANCREATE';
  ShipCreateTime           = 'SHIPCREATETIME';
  ShipCost                 = 'SHIPCOST';

TYPE
{
  Định nghĩa kiểu frame animation:
      Mỗi hướng có một bản ghi riêng
      Bản ghi ghi lại số frame animation và một mảng lưu chi tiết thông tin về frame
animation đó: số hiệu của frame, kiểu vẽ frame.
}
  TFrame   = Array[THeading] of Record
    NumFrames : Integer;
    Ani       : Array of Record
      FramePos   : Integer;
      FrameStyle : Integer;
    End;
  End;
{
```

Định nghĩa kiểu dữ liệu animator của một biến tàu. Ở đây có hai trường là Move và Attack nhưng tui mới dùng có trường Move thui, hè hè…
}
```
  TShipAni = Record
    Move,Attack : TFrame;
  End;
```

TYPE
{
Định nghĩa kiểu dữ liệu lưu trữ những thông tin cơ bản về một kiểu unit:
_HitPoint: Số máu mặc định của đơn vị.
_Damage: Số damage (tạm dịch: chỉ số sát thương) của đơn vị.
_AttackRange: Giới hạn tấn công của đơn vị.
_Speed: Tốc độ của đơn vị.
_Wait, _TurnWait: Thời gian chờ di chuyển và xoay đầu của đơn vị.
_Size, _Size2: Kích thước của đơn vị/kích thước chia hai của đơn vị, biến này lưu vì nhiều khi dùng biến này lém, đỡ phải tính lại.
_Gun: Kiểu súng của đơn vị.
_Explosion: Kiểu nổ của đơn vị (Hiện tại chưa để làm gì cả, gợi ý các bạn khai báo thêm vài unit kiểu nổ rùi thêm nó vào khi unit tiêu tùng giống như là thêm khói Burn khi tên lửa bắn đi vậy).
_NumOfImages: Số lượng ảnh của đơn vị.
_Images: Mảng lưu số hiệu các ảnh của đơn vị (Do AvenusHelper quản lý các ảnh theo số hiệu nên phải lưu trữ theo cách này để truy cập lại)
_ShipCanCreate: Tàu con mà unit này có thể sinh, nếu là NoneShip thì unit không có khả năng sinh tàu mới.
_DrawLevel: càng thấp thì được vẽ trước, cái này nhằm chỉnh lại các hành tinh không được vẽ sau các tàu chẳng hạn.
_BuildTime: là thời gian sinh ra loại đơn vị này.
_Cost: là chi phí để sinh loại đơn vị này.
}
```
  TUnitProperty = Record
    _HitPoint                       : Integer;
    _Damage                         : Integer;
    _AttackRange                    : Integer;
    _Speed                          : Integer;
    _Wait,_TurnWait                 : Integer;
    _Size,_Size2                    : Integer;
    _Gun                            : TGun;
    _Explosion                      : TSpaceShip;
    _NumOfImages                    : Integer;
    _Images                         : Array of Integer;
    _ShipCanCreate                  : TSpaceShip;
    _ShipAni                        : TShipAni;
    _DrawLevel                      : Byte;
    _BuildTime                      : Integer;
    _Cost                           : Integer;
  End;
```
{
Định nghĩa kiểu dữ liệu lưu trữ thông tin về một đơn vị:
Owner: Unit thuộc phe nào ?
HitPoint: Máu hiện tại của đơn vị.
Pos: Vị trí của đơn vị trên bản đồ.
Với đơn vị là FirePlanet:
Mức độ mờ, tốc độ mờ để vẽ viền của unit này.
Với các đơn vị tàu:
Dest: Đích đến của unit.

PatrolStart, PatrolDest: Điểm xuất phát và kết thúc của lệnh đi tuần (Patrol)
CurrentHead, Head: Hướng hiện tại, hướng cần xoay đến.
CurrentFrame: Frame ảnh hiện tại.
WaitTimeForMove: Số game frame cần để di chuyển.
UnitCmd, UnitNextCmd: Command của unit hiện tại và command kế tiếp.
UnitTarget, UnitNextTarget: Unit đích hiện tại và unit đích tiếp theo.
UnitGroup: Số hiệu nhóm của unit.
CreationCounting: Số bước thực hiện tạo tàu mới (Khi unit thực hiện lệnh tạo quân).
}

```
  TUnit = Record
    Owner                           : TPlayer;
    HitPoint                        : Integer;
    Pos                             : TRPoint;
    Case Typer : TSpaceShip of
      FirePlanet : (
        TransparentLevel : Integer;
        TransparentSpeed : Integer;
      );
      SupperBattleShip,
      BattleCruiseAmon,
      BattleCruiseBaton,
      CooNok,
      Scout,
      Interceptor : (
        Dest,PatrolStart,PatrolDest  : TRPoint;
        CurrentHead,Head             : THeading;
        CurrentFrame,
        WaitTimeForMove              : Integer;
        UnitCmd,UnitNextCmd          : TCommand;
        UnitTarget,UnitNextTarget    : TUnitCount;
        UnitGroup                    : Byte;
        CreationCounting             : Integer;
      );
  End;
```

{
Định nghĩa kiểu dữ liệu lưu thông tin cho từng player trong game:
Name: Dĩ nhiên là lưu…tên roài.
Money: Bi chừ em đang có bao nhiu xiền hỉ ?
CountUnit: Tổng số unit player này hiện đang có.
Enemy: Đánh dấu xem chú nào là địch ?
}

```
TYPE
  TPlayersData = Record
    Name         : String[20];
    Money        : Integer;
    CountUnit    : Integer;
    Enemy        : Array[TPlayer] of Boolean;
  End;
```

{
Định nghĩa kiểu dữ liệu lưu thông tin về bản đồ:
Scale: Kích cỡ zoom của bản đồ này.
ViewPos: Vị trí nhìn hiện tại.
Size: Kích cỡ của bản đồ, Size = Scale * DefaultMiniMapSize
StarBackGround: Ảnh nền, map này được dựng đơn giản với toàn các ảnh nền này xắp xếp bên cạnh nhau.
NumTile: Số lượng ảnh nhìn thấy được trên màn hình.
}

```
TYPE
```

```
  TWorld = Record
    Scale,ViewPos,Size : TPoint;
    StarBackGround     : Integer;
    NumTile            : TPoint;
  End;
{
  Đối tượng chính: GamePlay.
}
TYPE
  TGame = Class
{
  Đánh dấu game đã kết thúc.
}
    EndGame                      : Boolean;
{
  Đánh dấu game đang trong chế độ tạm dừng.
}
    PauseGame                    : Boolean;
{
  Dữ liệu về những người chơi trong game.
}
    GamePlayers                  : Array[TPlayer] of TPlayersData;
{
  Dữ liệu về tính năng chuẩn của các unit trong game.
}
    UnitsProperty                : Array[TSpaceShip] of TUnitProperty;
{
  Dữ liệu về các unit trong game.
}
    GameUnits                    : Array[TUnitCount] of TUnit;
{
  Dữ liệu về bản đồ trong game.
}
    GameWorld                    : TWorld;
{
  Dữ liệu về lệnh đang chờ đặt thực hiện (Bằng chuột - Giống như khi bạn ra lệnh Patrol
trong game Warcraft thì chương trình cần chờ bạn chọn vị trí bằng chuột mới thực hiện lệnh
điều khiển này).
}
    MouseCommand                 : TCommand;
{
  Vị trí đầu và vị trí cuối cùng người chơi kéo chuột (Khi chọn các đơn vị).
}
    SelectStart,SelectEnd        : TPoint;
{
  Phe mà người chơi điều khiển.
}
    HumanControl                 : TPlayer;
{
  Toạ độ unit dịch chuyển trong từng bước đi. Số lượng hướng càng nhìu thì unit xoay càng
mịn và tôi sử dụng toạ độ của unit kiểu số thực do nếu dùng số nguyên thì bước di chuyển
của unit sẽ có độ dài không đều nhau và khi chuyển lên toạ độ nguyên của màn hình unit sẽ
di chuyển trông giựt giựt kì lém á...
}
    Moving                       : Array[THeading,1..2] of Single;
{
```

Ảnh lưu bản đồ nhỏ, khi vẽ lại bản đồ nhỏ tôi chỉ vẽ lại ảnh này, còn việc cập nhật ảnh dùng bằng một hàm khác. Hàm này chỉ cập nhật bản đồ sau một khoảng thời gian cố định bởi nếu cập nhật liên tục thì tốc độc chương trình giảm thảm hại và kết quả sẽ là một đi không trở lại…
}
```
    MiniMapImage                : Integer;
```
{
Vị trí bản đồ được vẽ trên màn hình.
}
```
    MiniMapPosX,MiniMapPosY     : Integer;
```
{
Biến này lưu giữ trạng thái bản đồ có được thể hiện hay không, giúp người chơi có thể tắt thể hiện bản đồ nhằm tăng không gian điều khiển các unit của họ.
}
```
    MiniMapVisible              : Boolean;
```
{
Màu của các đơn vị khi xuất hiện trên bản đồ.
}
```
    PlayerColor                 : Array[TPlayer] of LongWord;
```
{
Đây là các thông tin tương tự như của minimap nhưng là của thành phần tôi gọi tạm là Panel, cái này thể hiện trạng thái của đơn vị đang được lựa chọn.
}
```
    PanelImage                  : Integer;
    PanelPosX,PanelPosY         : Integer;
    CommandPanelVisible         : Boolean;
```
{
Số hiệu đơn vị đầu tiên trong nhóm đơn vị được lựa chọn, khi chọn bạn có thể chọn tất cả các đơn vị của bạn trong khi thanh Panel chỉ có thể hiển thị thông tin về một đơn vị, đó chính là đơn vị này.
}
```
    FirstUnitSelect             : TUnitCount;
```
{
Số lượng frame game đã chạy (đếm theo số frame xử lý unit) và số lượng frame hình ảnh đã được cập nhật (số hình ảnh vẽ ra). Thông tin này dùng để tính GFPS (Game Frame Per Second) và SFPS (Screen Frame Per Second). Trong chương trình này tui đặt thời gian như trên nên SFPS đạt khoảng 28 FPS/1024x768x16bit color, số khung hình như thế này là chấp nhận được với game 2D (Hic, 3D mà thế này chắc giựt tung tròng mắt ra wé…hix).
}
```
    GameFrame,GameScreenFrame,
```
{
Thời điểm game bắt đầu.
Thời gian game đã chạy.
Thời điểm cuối cùng đã cập nhật màn hình.
Thời điểm cuối cùng đã cập nhật xử lý đơn vị.
Thời điểm cuối cùng đã cập nhật xử lý input (keyboard, mouse).
Thời điểm cuối cùng đã cập nhật minimap.
Thời điểm cuối cùng đã cập nhật tiền cho các phe.
}
```
    GameStart,GameTime,
    LastGameScreenTime,
    LastGameUnitTime,
    LastGameInputTime,
    LastGameMiniMapTime,
    LastGameMoneyTime           : LongWord;
```

```
{
```
Ảnh FirePlanet và quầng sáng bao quanh nó, hix, tại tôi ngại config cho kiểu đơn vị này nên làm kiểu lười, hì hì.
```
}
    GlobeImageNum,FlareImageNum  : Integer;
{
```
Lần cuối cùng sử dụng loại âm thanh (bắn, nổ). Đây là thông tin dùng để hạn chế việc gọi một âm thanh liên tục quá nhiều lần dẫn đến mất tài nguyên của máy tính. Đáng ra việc quản lý âm thanh nên tạo lập một class khác chuyên nghiệp hơn nhưng do ít thời gian quá nên gần như tôi bỏ qua việc xử lý âm thanh.
```
}
    LastMissileLauchSoundTime,
    LastMissileExploredSoundTime : LongWord;
{
```
Dữ liệu về âm thanh trong game.
```
}
    {$IfDef SoundPlaying}
    BackGroundMusic,
    MissileLauch,
    MissileExplored : Integer;
    {$EndIf}
{
```
Frame hiện tại dùng để tìm đích cho các unit ?
```
}
    IsFrameToFindTarget          : Boolean;
{
```
Thông tin debug:
      Hiển thị FPS, hiển thị thông tin debug ?
      Tổng số unit đang sử dụng ?
```
}
    ShowFPS,ShowDebug            : Boolean;
    UnitCounting                 : Integer;
{
```
Hằng số định nghĩa màu.
```
}
    White,Green,Gray,BlueGray    : LongWord;
{
```
Phương thức khởi tạo và huỷ lớp Game này.
```
}
    Constructor Create;
    Destructor Destroy; OverRide;
{
```
Hàm xử lý xuất nhập, xử lý các unit, xử lý các sự kiện và hàm chính gọi khi bắt đầu tiến trình xử lý một trận.
```
}
    Procedure ProcessInput;
    Procedure ProcessUnits;
    Procedure ProcessEvents;
    Procedure Play;
{
```
Đặt quan hệ giữa phe P1 và P2 là đối địch. Các đơn vị sẽ chỉ tìm và tấn công các đơn vị thuộc phe đối địch mà thôi (Dĩ nhiên !)
```
}
    Procedure SetPlayerEnemy(P1,P2 : TPlayer);
{
```
Cấu hình cho phe Player: Tên phe, số tiền có sẵn.

```
}
    Procedure SetupPlayer(Player : TPlayer;Name : String;Money : Integer);
{
```
Thêm tiền cho tất cả các phe, hàm này gọi khi chương trình phát sự kiện cập nhật tài nguyên cho các phe. Trong game này để đơn giản hoá việc xử lý tôi bỏ qua việc khai thác tài nguyên, sau một khoảng thời gian xác định game sẽ cung cấp cho mỗi phe một số tiền xác định.
```
}
    Procedure AddMoneyForAllPlayer;
{
```
Khởi tạo lại tất cả các biến đếm thời gian, đếm frame…
```
}
    Procedure RestartCount;
{
```
Đọc các cấu hình của các loại đơn vị trong game (từ file text).
```
}
    Procedure LoadSetting;
{
```
Tạo một thông điệp đến người chơi.
```
}
    Procedure CreateMessage(Msg : String);
{
```
Hàm xử lý các sự kiện chuột được nhấn, thả, điều khiển từ người chơi…
```
}
    Procedure ProceedMouseLHolding;
    Procedure ProceedMouseRHolding;
    Procedure ProceedMouseLReleased;
    Procedure ProceedMouseRReleased;
    Procedure ProceedPlayerInput;
{
```
Người chơi đặt lệnh dừng cho tất cả các đơn vị họ đang chọn.
```
}
    Procedure HumanSetCommandStop;
{
```
Người chơi đặt lệnh sinh quân cho tất cả các đơn vị họ đang chọn. Vì mỗi đơn vị chỉ có thể sinh ra một loại quân nhất định nên không cần biết yêu cầu là sinh đơn vị gì.
```
}
    Procedure HumanSetCommandCreate;
{
```
Người chơi đặt lệnh cho nhóm quân đang chọn di chuyển đến tọa độ [X,Y] trên map.
```
}
    Procedure HumanSetCommandMove(X,Y : Integer);
{
```
Người chơi đặt lệnh cho nhóm quân đang chọn tấn công unit số hiệu Target.
```
}
    Procedure HumanSetCommandAttack(Target : TUnitCount);
{
```
Người chơi đặt lệnh cho nhóm quân đang chọn tấn công đến vị trí [X,Y] trên map, trong quá trình di chuyển unit sẽ tấn công bất cứ mục tiêu nào nó gặp mặt.
```
}
    Procedure HumanSetCommandAttackTo(X,Y : Integer);
{
```
Người chơi đặt lệnh cho nhóm quân đang chọn đi tuần đến vị trí [X,Y] trên map, trong quá trình di chuyển unit sẽ tấn công bất cứ mục tiêu nào nó nhìn thấy.
```
}
    Procedure HumanSetCommandPatrol(X,Y : Integer);
```

```
{
  Cấu hình ảnh World (Hè, thực chất chỉ là load ảnh nền liên quan đến WorldMap thôi).
}
    Procedure SetupWorldImage;
{
  Cấu hình kích thước bản đồ, kích thước bản đồ bằng:
    [ScaleSizeX * MiniMapSizeX , ScaleSizeY * MiniMapSizeY]
}
    Procedure SetupWorldSize(ScaleSizeX,ScaleSizeY : Integer);
{
  Đặt vị trí nhìn trên bản đồ ở [ViewX , ViewY]
}
    Procedure WorldViewAt(ViewX,ViewY : Integer);
{
  Đọc bản đồ từ file, cái này tui…chưa làm, mới chỉ đặt một bản đồ linh tinh và mấy unit để
test thôi, tôi nghĩ cái này các bạn có thể tự làm được một cách dễ dàng.
}
    Procedure LoadWorld(FileName : String);
{
  Cập nhật MiniMap
}
    Procedure WorldMiniMapUpdate;
{
  Đọc cấu hình của unit từ file.
}
    Procedure SetupUnitsSetting;
{
  Cấu hình hình ảnh của các unit không được cấu hình bằng file (FirePlanet).
}
    Procedure SetupUnitsImage;
{
  Lấy cấu hình unit từ file, tên và các cấu hình của unit đã có đủ trong file cấu hình.
}
    Procedure ProceedUnitSettingFile(FileName : String);
{
  Tạo một unit UnitTyper mới của Player ở vị trí [PosX , PosY], hàm trả về số hiệu của unit
nếu tạo lập thành công, còn nếu không còn vị trí trống để sử dụng cho unit này thì hàm trả
về Low(TunitCount) như vậy đơn vị số hiệu này không bao giờ được sử dụng.
}
    Function  CreateNewUnit(UnitTyper : TSpaceShip;
                            Player : TPlayer;
                            PosX,PosY : Integer) : TUnitCount;
{
  Huỷ unit số hiệu UnitNum.
}
    Function  DestroyUnit(UnitNum : TUnitCount) : Boolean;
{
  Khởi tạo dữ liệu chuẩn của kiểu UnitTyper cho unit số hiệu UnitNum.
}
    Procedure SetupUnit(UnitNum : TUnitCount;
                        UnitTyper : TSpaceShip;
                        Player : TPlayer);
{
  "Thanh toán xiền" khi sinh unit mới, nếu Player không còn đủ tài nguyên, hàm trả về False
còn nếu đủ sẽ trừ đi số tài nguyên tương ứng và trả về giá trị True.
```

```
}
    Function  CostForNewUnit(Player : TPlayer;
                             UnitTyper : TSpaceShip) : Boolean;
{
  Đặt unit số hiệu UnitNum được đưa vào nhóm chọn.
}
    Procedure SelectUnit(UnitNum : TUnitCount);
{
  Bỏ chọn tất cả các unit.
}
    Procedure UnSelectAllUnit;
{
  Lấy unit trên cùng được chọn từ toạ độ [MX,MY]. Tham số HumanOwner để chỉ đơn vị
được chọn chỉ được phép thuộc nhóm unit của người chơi. Hàm trả về số hiệu của unit tìm
thấy, còn nếu không thấy sẽ trả về giá trị Low(TunitCount).
}
    Function  GetUnitAtMouse(MX,MY : Integer;
                             HumanOwner : Boolean) : TUnitCount;
{
  Hàm chọn các unit bằng chuột, tham số Add cho biết có thêm unit vào nhóm đang chọn
hay xoá bỏ tất cả và chọn lại từ đầu.
}
    Procedure CallMouseSelectionUnit(Add : Boolean);
{
  Hàm này cho biết unit cần xoay về hướng nào để đến được đích của nó. Ở đây chỉ đơn
giản là xoay về hướng có toạ độ thẳng nhất.
}
    Function  UnitGetBestHeading(UnitNum : TUnitCount) : THeading;
{
  Khoảng các giữa unit và đích nhỏ hơn Long ?
}
    Function  UnitCloseToTarget(UnitNum : TUnitCount;
                                Long : Integer) : Boolean;
{
  Chỉnh hướng unit về phía đích của nó.
}
    Procedure UnitPointToTarget(UnitNum : TUnitCount);
{
  Đặt lệnh cho unit "bắn" vào đích. Khi gọi lệnh này, unit sẽ sinh loại Missile tương ứng
nhắm đến đích của nó.
}
    Procedure UnitFire(UnitNum : TUnitCount);
{
  Tính thiệt hại khi unit chạm đích (đây là dành cho các unit loại Missile).
}
    Procedure UnitHit(UnitNum : TUnitCount);
{
  Giảm máu của unit đi Damage đơn vị.
}
    Procedure UnitDecHitPoint(UnitNum : TUnitCount;Damage : Integer);
{
  Unit có trong tầm nhìn không ?
}
    Function  UnitVisible(UnitNum : TUnitCount) : Boolean;
{
```

```
  Hướng cạnh StartHead gần với EndHead nhất.
}
    Function  HeadNear(StartHead,EndHead : THeading) : THeading;
{
  Lấy một điểm bất kì trong khoảng cách Leng đến toạ độ [X,Y]
}
    Procedure PointNear(X,Y,Leng : Integer;Var DX,DY : Integer);
{
  Độ dài từ [X1,Y1] đến [X2,Y2].
}
    Function  Long(X1,Y1,X2,Y2 : Single) : Single;
{
  Sinh một số hiệu phe bất kì.
}
    Function  RandomPlayer : TPlayer;
{
  Lấy lệnh kế tiếp cho unit: Khi người chơi đặt lệnh cho unit mà unit đang thực thi một lệnh
khác, lệnh đặt vào UnitNextCmd và hàm này thực hiện kiểm tra UnitNextCmd để chuyển
lệnh của unit sang lệnh mới.
}
    Function  UnitGetNextCmd(UnitNum : TUnitCount) : Boolean;
{
  Chuyển lệnh của unit về lệnh trước đó nó đang nhận ( Nếu unit đang nhận lệnh
AttackPatrol thì lệnh trước đó là Patrol, nếu unit đang nhận lệnh AttackMove thì lệnh trước
đó là AttackTo).
}
    Procedure UnitResetCommand(UnitNum : TUnitCount);
{
  Đặt lệnh tạo quân cho unit.
}
    Procedure UnitSetCreate(UnitNum : TUnitCount);
{
  Đặt lệnh cho unit di chuyển đến vị trí [DestX,DestY].
}
    Procedure UnitSetMove(UnitNum : TUnitCount;DestX,DestY : Integer);
{
  Đặt lệnh cho unit đi tuần đến vị trí [DestX,DestY].
}
    Procedure UnitSetPatrol(UnitNum : TUnitCount;DestX,DestY : Integer);
{
  Đặt lệnh cho unit tấn công đến vị trí [DestX,DestY].
}
    Procedure UnitSetAttackTo(UnitNum : TUnitCount;DestX,DestY : Integer);
{
  Đặt lệnh cho unit tấn công Target.
}
    Procedure UnitSetAttack(UnitNum : TUnitCount;Target : Integer);
{
  Đặt lệnh cho unit đang trong trạng thái đi tuần tấn công unit Target.
}
    Procedure UnitSetAttackPatrol(UnitNum : TUnitCount;Target : Integer);
{
  Đặt lệnh cho unit đang trong trạng thái di chuyển tấn công unit Target.
}
    Procedure UnitSetAttackMove(UnitNum : TUnitCount;Target : Integer);
```

```
{
  Unit tìm đối phương trong tầm tấn công (Tìm đối phương gần nhất).
}
    Function  UnitFindTarget(UnitNum : TUnitCount) : TUnitCount;
{
  Tìm unit ở gần UnitNum, hàm này xây dựng để xử dụng khi dãn đội hình ra.
}
    Function  UnitFindNear(UnitNum : TUnitCount) : TUnitCount;
{
  Xử lý các unit đặc biệt (FirePlanet).
}
    Procedure ProcessSpecifyUnit(UnitNum : TUnitCount);
{
  Xử lý các unit bình thường.
}
    Procedure ProcessNormalUnit(UnitNum : TUnitCount);
{
  Xử lý unit đang không có lệnh.
}
    Procedure ProceedUnitStand(UnitNum : TUnitCount);
{
  Xử lý unit đang nhận lệnh Move.
}
    Procedure ProceedUnitMove(UnitNum : TUnitCount);
{
  Xử lý unit đang nhận lệnh Attack.
}
    Procedure ProceedUnitAttack(UnitNum : TUnitCount);
{
  Xử lý unit đang nhận lệnh AttackP (Attack while patrol).
}
    Procedure ProceedUnitAttackP(UnitNum : TUnitCount);
{
  Xử lý unit đang nhận lệnh AttackT (Attack to somewhere).
}
    Procedure ProceedUnitAttackT(UnitNum : TUnitCount);
{
  Xử lý unit đang nhận lệnh AttackM (Attack while move to somewhere).
}
    Procedure ProceedUnitAttackM(UnitNum : TUnitCount);
{
  Xử lý unit đang nhận lệnh Patrol.
}
    Procedure ProceedUnitPatrol(UnitNum : TUnitCount);
{
  Xử lý unit đang nhận lệnh Burn.
}
    Procedure ProceedUnitBurning(UnitNum : TUnitCount);
{
  Vẽ nền bản đồ.
}
    Procedure RenderWorld;
{
  Vẽ unit UnitNum.
}
```

```
    Procedure RenderUnits(UnitNum : TUnitCount); OverLoad;
{
  Vẽ tất cả các unit.
}
    Procedure RenderUnits; OverLoad;
{
  Vẽ các MiniMap, Panel, các thông tin khác...
}
    Procedure RenderInfos;
{
  Vẽ toàn bộ.
}
    Procedure RenderScene;
{
  Lấy loại tàu bằng tên.
}
    Function  GetShipTyper(Name : String) : TSpaceShip;
{
  Lấy số hiệu phe bằng tên.
}
    Function  GetPlayer(Name : String) : TPlayer;
{
  Lấy hướng bằng tên.
}
    Function  GetHeading(Name : String) : THeading;
{
  Lấy loại súng bằng tên.
}
    Function  GetGun(Name : String) : TGun;
  End;

VAR
  MyGame : TGame;

IMPLEMENTATION

USES CommonFuncs;

CONSTRUCTOR TGame.Create;
  Begin
    //Moving generating
    Moving[H01,1]:=+0;
    Moving[H01,2]:=-1;
    Moving[H02,1]:=+Abs(Sin(DegToRad(90/8*1)));
    Moving[H02,2]:=-Abs(Cos(DegToRad(90/8*1)));
    Moving[H03,1]:=+Abs(Sin(DegToRad(90/8*2)));
    Moving[H03,2]:=-Abs(Cos(DegToRad(90/8*2)));
    Moving[H04,1]:=+Abs(Sin(DegToRad(90/8*3)));
    Moving[H04,2]:=-Abs(Cos(DegToRad(90/8*3)));
    Moving[H05,1]:=+Abs(Sin(DegToRad(90/8*4)));
    Moving[H05,2]:=-Abs(Cos(DegToRad(90/8*4)));
    Moving[H06,1]:=+Abs(Sin(DegToRad(90/8*5)));
    Moving[H06,2]:=-Abs(Cos(DegToRad(90/8*5)));
    Moving[H07,1]:=+Abs(Sin(DegToRad(90/8*6)));
    Moving[H07,2]:=-Abs(Cos(DegToRad(90/8*6)));
    Moving[H08,1]:=+Abs(Sin(DegToRad(90/8*7)));
    Moving[H08,2]:=-Abs(Cos(DegToRad(90/8*7)));
    Moving[H09,1]:=+1;
    Moving[H09,2]:=+0;
    Moving[H10,1]:=+Abs(Sin(DegToRad(90/8*7)));
```

```
Moving[H10,2]:=+Abs(Cos(DegToRad(90/8*7)));
Moving[H11,1]:=+Abs(Sin(DegToRad(90/8*6)));
Moving[H11,2]:=+Abs(Cos(DegToRad(90/8*6)));
Moving[H12,1]:=+Abs(Sin(DegToRad(90/8*5)));
Moving[H12,2]:=+Abs(Cos(DegToRad(90/8*5)));
Moving[H13,1]:=+Abs(Sin(DegToRad(90/8*4)));
Moving[H13,2]:=+Abs(Cos(DegToRad(90/8*4)));
Moving[H14,1]:=+Abs(Sin(DegToRad(90/8*3)));
Moving[H14,2]:=+Abs(Cos(DegToRad(90/8*3)));
Moving[H15,1]:=+Abs(Sin(DegToRad(90/8*2)));
Moving[H15,2]:=+Abs(Cos(DegToRad(90/8*2)));
Moving[H16,1]:=+Abs(Sin(DegToRad(90/8*1)));
Moving[H16,2]:=+Abs(Cos(DegToRad(90/8*1)));
Moving[H17,1]:=+0;
Moving[H17,2]:=+1;
Moving[H18,1]:=-Abs(Sin(DegToRad(90/8*1)));
Moving[H18,2]:=+Abs(Cos(DegToRad(90/8*1)));
Moving[H19,1]:=-Abs(Sin(DegToRad(90/8*2)));
Moving[H19,2]:=+Abs(Cos(DegToRad(90/8*2)));
Moving[H20,1]:=-Abs(Sin(DegToRad(90/8*3)));
Moving[H20,2]:=+Abs(Cos(DegToRad(90/8*3)));
Moving[H21,1]:=-Abs(Sin(DegToRad(90/8*4)));
Moving[H21,2]:=+Abs(Cos(DegToRad(90/8*4)));
Moving[H22,1]:=-Abs(Sin(DegToRad(90/8*5)));
Moving[H22,2]:=+Abs(Cos(DegToRad(90/8*5)));
Moving[H23,1]:=-Abs(Sin(DegToRad(90/8*6)));
Moving[H23,2]:=+Abs(Cos(DegToRad(90/8*6)));
Moving[H24,1]:=-Abs(Sin(DegToRad(90/8*7)));
Moving[H24,2]:=+Abs(Cos(DegToRad(90/8*7)));
Moving[H25,1]:=-1;
Moving[H25,2]:=+0;
Moving[H26,1]:=-Abs(Sin(DegToRad(90/8*7)));
Moving[H26,2]:=-Abs(Cos(DegToRad(90/8*7)));
Moving[H27,1]:=-Abs(Sin(DegToRad(90/8*6)));
Moving[H27,2]:=-Abs(Cos(DegToRad(90/8*6)));
Moving[H28,1]:=-Abs(Sin(DegToRad(90/8*5)));
Moving[H28,2]:=-Abs(Cos(DegToRad(90/8*5)));
Moving[H29,1]:=-Abs(Sin(DegToRad(90/8*4)));
Moving[H29,2]:=-Abs(Cos(DegToRad(90/8*4)));
Moving[H30,1]:=-Abs(Sin(DegToRad(90/8*3)));
Moving[H30,2]:=-Abs(Cos(DegToRad(90/8*3)));
Moving[H31,1]:=-Abs(Sin(DegToRad(90/8*2)));
Moving[H31,2]:=-Abs(Cos(DegToRad(90/8*2)));
Moving[H32,1]:=-Abs(Sin(DegToRad(90/8*1)));
Moving[H32,2]:=-Abs(Cos(DegToRad(90/8*1)));
//Debug and condition setting
EndGame:=False;
PauseGame:=False;
ShowFPS:=False;
ShowDebug:=False;
UnitCounting:=0;
FirstUnitSelect:=Low(TUnitCount);
//
{$IfDef FullScreen}
CreateFullScreen(ScrWidth,ScrHeight,BD16Bit,Caption);
{$Else}
CreateWindow(ScrWidth,ScrHeight,Caption);
{$EndIf}
LoadFont(DataPath+'Font.png',16,20,8,16,$0);
SetScreenBPP(32);
//Mini map setting
MiniMapVisible:=True;
MiniMapImage:=CreateImage(MiniMapSizeX,MiniMapSizeY,FormatA1R5G5B5);
MiniMapPosX:=ScrWidth-MiniMapSizeX-10;
```

```
    MiniMapPosY:=10;
    //Panel setting
    CommandPanelVisible:=True;
    PanelImage:=CreateImage(PanelSizeX,PanelSizeY,FormatA1R5G5B5);
    PanelPosX:=10;
    PanelPosY:=ScrHeight-PanelSizeY-10;
    //
    SetupWorldImage;
    SetupUnitsImage;
    LoadSetting;
    //Colour setup
    White:=RGB2LongWord(255,255,255);
    Green:=RGB2LongWord(000,255,000);
    Gray:=RGB2LongWord(40,40,40);
    BlueGray:=RGB2LongWord(100,100,165);
    //Player color setting
    PlayerColor[PlayerNature]:=RGB2LongWord(255,255,255);//White
    PlayerColor[Player1]:=Color32To15(RGB2LongWord(255,000,000));//Red
    PlayerColor[Player2]:=Color32To15(RGB2LongWord(000,255,000));//Green
    PlayerColor[Player3]:=Color32To15(RGB2LongWord(000,000,255));//Blue
    PlayerColor[Player4]:=Color32To15(RGB2LongWord(255,255,128));//Yellow
    //
    //Sound setting
    //
    {$IfDef SoundPlaying}
    BackGroundMusic:=CreateSound(DataPath+'Sounds\BackGround.Mp3',False);
    MissileLauch:=CreateSound(DataPath+'Sounds\MissileLauch.Wav',False);
    MissileExplored:=CreateSound(DataPath+'Sounds\MissileExplored.Wav',False);
    {$EndIf}
  End;

DESTRUCTOR TGame.Destroy;
  Begin
    DestroyScreen;
  End;

PROCEDURE TGame.ProceedMouseLHolding;
  Begin
    If MouseCommand=MouseSelection then
      Begin
        SelectEnd.X:=MouseX;
        SelectEnd.Y:=MouseY;
      End
    Else
      Begin
        //Click on command panel ?
        If CommandPanelVisible then
          Begin

          End;
        //Click on mini map ?
        If MiniMapVisible then
          Begin
            If (MouseX>=MiniMapPosX) and
               (MouseX<=MiniMapPosX+MiniMapSizeX) and
               (MouseY>=MiniMapPosY) and
               (MouseY<=MiniMapPosY+MiniMapSizeY) then
              Begin
                WorldViewAt((MouseX-MiniMapPosX)*GameWorld.Scale.X,
                            (MouseY-MiniMapPosY)*GameWorld.Scale.Y);
                Exit;
              End;
          End;
        If MouseCommand=NoCmd then
```

```
          Begin
            MouseCommand:=MouseSelection;
            SelectStart.X:=MouseX;
            SelectStart.Y:=MouseY;
            SelectEnd.X:=MouseX;
            SelectEnd.Y:=MouseY;
          End;
      End;
  End;

PROCEDURE TGame.ProceedMouseRHolding;
  Begin
  End;

PROCEDURE TGame.ProceedMouseLReleased;
  Var UnitNum   : TUnitCount;
      OnMiniMap : Boolean;
      X,Y       : Integer;
  Begin
    OnMiniMap:=False;
    X:=0;Y:=0;
    If MiniMapVisible then
      Begin
        If (MouseX>=MiniMapPosX) and
           (MouseX<=MiniMapPosX+MiniMapSizeX) and
           (MouseY>=MiniMapPosY) and
           (MouseY<=MiniMapPosY+MiniMapSizeY) then
          Begin
            OnMiniMap:=True;
            X:=(MouseX-MiniMapPosX)*GameWorld.Scale.X;
            Y:=(MouseY-MiniMapPosY)*GameWorld.Scale.Y;
          End
      End;
    Case MouseCommand of
      MouseSelection :
        Begin
          CallMouseSelectionUnit(KeyDown(Key_LShift) or
                                 KeyDown(Key_LControl));
        End;
      CmdAttack :
        Begin
          If OnMiniMap then
            Begin
              HumanSetCommandAttackTo(X,Y);
            End
          Else
            Begin
              UnitNum:=GetUnitAtMouse(MouseX,MouseY,False);
              If UnitNum=Low(TUnitCount) then
                Begin
                  HumanSetCommandAttackTo(MouseX+GameWorld.ViewPos.X,
                                          MouseY+GameWorld.ViewPos.Y);
                End
              Else
                Begin
                  HumanSetCommandAttack(UnitNum);
                End;
            End;
        End;
      CmdPatrol :
        Begin
          If OnMiniMap then
            Begin
              HumanSetCommandPatrol(X,Y);
```

```
            End
          Else
            Begin
              HumanSetCommandPatrol(MouseX+GameWorld.ViewPos.X,
                                    MouseY+GameWorld.ViewPos.Y);
            End;
        End;
    End;
    MouseCommand:=NoCmd;
  End;

PROCEDURE TGame.ProceedMouseRReleased;
  Var UnitNum : TUnitCount;
  Begin
    If MouseCommand=NoCmd then
      Begin
        //Click on mini map ?
        If MiniMapVisible then
          Begin
            If (MouseX>=MiniMapPosX) and
               (MouseX<=MiniMapPosX+MiniMapSizeX) and
               (MouseY>=MiniMapPosY) and
               (MouseY<=MiniMapPosY+MiniMapSizeY) then
              Begin
                HumanSetCommandMove((MouseX-MiniMapPosX)*GameWorld.Scale.X,
                                    (MouseY-MiniMapPosY)*GameWorld.Scale.Y);
                Exit;
              End;
          End;
        //Other wise
        UnitNum:=GetUnitAtMouse(MouseX,MouseY,False);
        If UnitNum=Low(TUnitCount) then
          Begin
            HumanSetCommandMove(MouseX+GameWorld.ViewPos.X,
                                MouseY+GameWorld.ViewPos.Y);
          End
        Else
          Begin
            If GameUnits[UnitNum].Owner=HumanControl then
              Begin
                HumanSetCommandMove(MouseX+GameWorld.ViewPos.X,
                                    MouseY+GameWorld.ViewPos.Y);
              End
            Else
              Begin
                HumanSetCommandAttack(UnitNum);
              End;
          End;
      End
    Else
      Begin
        //Clear waiting command
        MouseCommand:=NoCmd;
      End;
  End;

PROCEDURE TGame.ProceedPlayerInput;
  Begin
    //LeftAlt+X ?
    If KeyDown(Key_LAlt) and
       KeyPress(Key_X) then EndGame:=True;
    //Arrow key: Take map view scroll
    If KeyDown(Key_Up) then
      WorldViewAt(Round(GameWorld.ViewPos.X),
```

```
                  Round(GameWorld.ViewPos.Y)-ScrollSpeed);
    If KeyDown(Key_Down) then
      WorldViewAt(Round(GameWorld.ViewPos.X),
                  Round(GameWorld.ViewPos.Y)+ScrollSpeed);
    If KeyDown(Key_Left) then
      WorldViewAt(Round(GameWorld.ViewPos.X)-ScrollSpeed,
                  Round(GameWorld.ViewPos.Y));
    If KeyDown(Key_Right) then
      WorldViewAt(Round(GameWorld.ViewPos.X)+ScrollSpeed,
                  Round(GameWorld.ViewPos.Y));
    //Static HotKey
    If KeyPress(Key_A) then
      MouseCommand:=CmdAttack;
    If KeyPress(Key_P) then
      MouseCommand:=CmdPatrol;
    If KeyPress(Key_S) then
      HumanSetCommandStop;
    If KeyPress(Key_C) then
      HumanSetCommandCreate;
    If KeyPress(Key_Space) then
      PauseGame:=Not PauseGame;
    //Mouse release
    If MouseReleasedL then
      ProceedMouseLReleased;
    If MouseReleasedR then
      ProceedMouseRReleased;
    //Toggle mini map visible
    If KeyPress(Key_Capslock) then
      MiniMapVisible:=Not MiniMapVisible;
    //Toggle command panel visible
    If KeyPress(Key_Tab) then
      CommandPanelVisible:=Not CommandPanelVisible;
    //Debug control Left Ctrl+P
    If KeyDown(Key_LControl) and
       KeyPress(Key_P) then ShowFPS:=Not ShowFPS;
    If KeyDown(Key_LControl) and
       KeyPress(Key_D) then ShowDebug:=Not ShowDebug;
  End;

PROCEDURE TGame.ProcessInput;
  Begin
    If GameTime-LastGameInputTime>InputUpdate then
      LastGameInputTime:=GameTime
    Else Exit;
    GetInputStatus;
    If EventMessage=False then EndGame:=True;
    If MouseHoldL then ProceedMouseLHolding
    Else
    If MouseHoldR then ProceedMouseRHolding
    Else
      ProceedPlayerInput;
  End;

PROCEDURE TGame.ProcessUnits;
  Var Z : TUnitCount;
  Begin
    If Not PauseGame then
      Begin
        If GameTime-LastGameUnitTime>UnitUpdate then
          Begin
            Inc(GameFrame);
            LastGameUnitTime:=GameTime;
          End
        Else Exit;
```

```
        For Z:=Low(TUnitCount) to
               High(TUnitCount) do
          With GameUnits[Z] do
            If HitPoint>0 then
              Begin
                Case Typer of
                  FirePlanet :
                    Begin
                      TransparentLevel:=TransparentLevel+TransparentSpeed;
                      If TransparentLevel<64 then
                        Begin
                          TransparentLevel:=64;
                          TransparentSpeed:=-TransparentSpeed;
                        End;
                      If TransparentLevel>255 then
                        Begin
                          TransparentLevel:=255;
                          TransparentSpeed:=-TransparentSpeed;
                        End;
                    End
                  Else
                    ProcessNormalUnit(Z);
                End;
              End;
      End;
  End;

PROCEDURE TGame.ProcessEvents;
  Begin
    //Update mini map
    If GameTime-LastGameMiniMapTime>MiniMapUpdate then
      Begin
        LastGameMiniMapTime:=GameTime;
        WorldMiniMapUpdate;
      End;
    //Update money
    If GameTime-LastGameMoneyTime>MoneyTime then
      Begin
        LastGameMoneyTime:=GameTime;
        AddMoneyForAllPlayer;
      End;
    IsFrameToFindTarget:=GameFrame mod FrameRateForFindTarget=0;
    GameTime:=MMSystem.TimeGetTime;
  End;

PROCEDURE TGame.Play;
  Begin
    LoadWorld('Demo.World');
    RestartCount;
    {$IfDef SoundPlaying}
    SoundPlay(BackGroundMusic);
    {$EndIf}
    Repeat
      ProcessInput;
      ProcessUnits;
      ProcessEvents;
      RenderScene;
    Until EndGame;
  End;

PROCEDURE TGame.HumanSetCommandStop;
  Var Z : TUnitCount;
  Begin
    For Z:=Low(TUnitCount) to
```

```
            High(TUnitCount) do
        //Unit is owner
        If (GameUnits[Z].Owner=HumanControl) and
        //Unit alive
           (GameUnits[Z].HitPoint>0) and
        //Unit selected
           (GameUnits[Z].UnitGroup and 128=128) then
           Begin
             //Set command moving !
             UnitResetCommand(Z);
           End;
  End;

PROCEDURE TGame.HumanSetCommandCreate;
  Var Z : TUnitCount;
  Begin
    For Z:=Low(TUnitCount) to
            High(TUnitCount) do
        //Unit is owner
        If (GameUnits[Z].Owner=HumanControl) and
        //Unit alive
           (GameUnits[Z].HitPoint>0) and
        //Unit selected
           (GameUnits[Z].UnitGroup and 128=128) then
           Begin
             //Set command moving !
             UnitSetCreate(Z);
           End;
  End;

PROCEDURE TGame.HumanSetCommandMove(X,Y : Integer);
  Var Z : TUnitCount;
  Begin
    For Z:=Low(TUnitCount) to
            High(TUnitCount) do
        //Unit is owner
        If (GameUnits[Z].Owner=HumanControl) and
        //Unit alive
           (GameUnits[Z].HitPoint>0) and
        //Unit selected
           (GameUnits[Z].UnitGroup and 128=128) then
           Begin
             //Set command moving !
             UnitSetMove(Z,X+Random(AroundLeng)-AroundLeng ShR 1,
                           Y+Random(AroundLeng)-AroundLeng ShR 1);
           End;
  End;

PROCEDURE TGame.HumanSetCommandAttack(Target : TUnitCount);
  Var Z : TUnitCount;
  Begin
    For Z:=Low(TUnitCount) to
            High(TUnitCount) do
        //Unit is owner
        If (GameUnits[Z].Owner=HumanControl) and
        //Unit alive
           (GameUnits[Z].HitPoint>0) and
        //Unit selected
           (GameUnits[Z].UnitGroup and 128=128) then
           Begin
             //Set command attack !
             UnitSetAttack(Z,Target);
           End;
  End;
```

```
PROCEDURE TGame.HumanSetCommandAttackTo(X,Y : Integer);
  Var Z : TUnitCount;
  Begin
    For Z:=Low(TUnitCount) to
           High(TUnitCount) do
      //Unit is owner
      If (GameUnits[Z].Owner=HumanControl) and
      //Unit alive
         (GameUnits[Z].HitPoint>0) and
      //Unit selected
         (GameUnits[Z].UnitGroup and 128=128) then
         Begin
           //Set command attack to point !
           UnitSetAttackTo(Z,X+Random(AroundLeng)-AroundLeng ShR 1,
                           Y+Random(AroundLeng)-AroundLeng ShR 1);
         End;
  End;

PROCEDURE TGame.HumanSetCommandPatrol(X,Y : Integer);
  Var Z : TUnitCount;
  Begin
    For Z:=Low(TUnitCount) to
           High(TUnitCount) do
      //Unit is owner
      If (GameUnits[Z].Owner=HumanControl) and
      //Unit alive
         (GameUnits[Z].HitPoint>0) and
      //Unit selected
         (GameUnits[Z].UnitGroup and 128=128) then
         Begin
           //Set command patrol !
           UnitSetPatrol(Z,X+Random(AroundLeng)-AroundLeng ShR 1,
                         Y+Random(AroundLeng)-AroundLeng ShR 1);
         End;
  End;

PROCEDURE TGame.SetPlayerEnemy(P1,P2 : TPlayer);
  Begin
    GamePlayers[P1].Enemy[P2]:=True;
    GamePlayers[P2].Enemy[P1]:=True;
  End;

PROCEDURE TGame.SetupPlayer(Player : TPlayer;Name : String;Money : Integer);
  Begin
    GamePlayers[Player].Name:=Name;
    GamePlayers[Player].Money:=Money;
    GamePlayers[Player].CountUnit:=0;
  End;

PROCEDURE TGame.AddMoneyForAllPlayer;
  Var Player : TPlayer;
  Begin
    For Player:=Low(TPlayer) to
                High(TPlayer) do
      GamePlayers[Player].Money:=DefaultMoney;
  End;

PROCEDURE TGame.RestartCount;
  Begin
    GameStart:=MMSystem.TimeGetTime;
    LastGameScreenTime:=GameStart;
    LastGameUnitTime:=GameStart;
    LastGameInputTime:=GameStart;
```

```
    LastGameMoneyTime:=GameStart;
    GameFrame:=0;
    GameScreenFrame:=0;
  End;

PROCEDURE TGame.LoadSetting;
  Begin
    SetupUnitsSetting;
    With UnitsProperty[FirePlanet] do
      Begin
        _HitPoint:=1;
      End;
  End;

PROCEDURE TGame.CreateMessage(Msg : String);
  Begin
  End;

PROCEDURE TGame.SetupWorldImage;
  Begin
    With GameWorld do
      Begin
        StarBackGround:=LoadImage(DataPath+'Stars.Jpg',FormatR5G6B5);
      End;
  End;

PROCEDURE TGame.SetupWorldSize(ScaleSizeX,ScaleSizeY : Integer);
  Begin
    With GameWorld do
      Begin
        //Scale out of range ?
        If ScaleSizeX>High(Integer) div MiniMapSizeX then
          ScaleSizeX:=High(Integer) div MiniMapSizeX;
        If ScaleSizeY>High(Integer) div MiniMapSizeY then
          ScaleSizeY:=High(Integer) div MiniMapSizeY;
        Scale.X:=ScaleSizeX;
        Scale.Y:=ScaleSizeY;
        Size.X:=Scale.X*MiniMapSizeX;
        Size.Y:=Scale.Y*MiniMapSizeY;
        NumTile.X:=ScrWidth div GetImageWidth(StarBackGround)+1;
        NumTile.Y:=ScrHeight div GetImageHeight(StarBackGround)+1;
      End;
  End;

PROCEDURE TGame.WorldViewAt(ViewX,ViewY : Integer);
  Begin
    //Cal map view
    If ViewX<0 then ViewX:=0;
    If ViewY<0 then ViewY:=0;
    If ViewX+ScrWidth>GameWorld.Size.X then
      ViewX:=GameWorld.Size.X-ScrWidth;
    If ViewY+ScrHeight>GameWorld.Size.Y then
      ViewY:=GameWorld.Size.Y-ScrHeight;
    GameWorld.ViewPos.X:=ViewX;
    GameWorld.ViewPos.Y:=ViewY;
  End;

PROCEDURE TGame.LoadWorld(FileName : String);
  Var Z : Integer;
      P : TPlayer;
  Begin
    For P:=Low(TPlayer) to
           High(TPlayer) do
      Begin
```

```
        FillChar(GamePlayers[P].Enemy,
                 SizeOf(GamePlayers[P].Enemy),True);
        GamePlayers[P].Enemy[P]:=False;
        GamePlayers[P].Enemy[NonePlayer]:=False;
        GamePlayers[P].Enemy[PlayerNature]:=False;
      End;
    HumanControl:=Player1;
    SetupPlayer(Player1,'',DefaultMoney*3);
    SetupPlayer(Player2,'',DefaultMoney*3);
    SetupPlayer(Player3,'',DefaultMoney*3);
    SetupPlayer(Player4,'',DefaultMoney*3);
    SetupPlayer(HumanControl,'',DefaultMoney);
    //Creating a map size 16000x16000 pixel
    SetupWorldSize(80,80);
    Randomize;
    CreateNewUnit(FirePlanet,PlayerNature,
                  Random(GameWorld.Size.X),Random(GameWorld.Size.Y));
    CreateNewUnit(FirePlanet,PlayerNature,
                  Random(GameWorld.Size.X),Random(GameWorld.Size.Y));
    CreateNewUnit(FirePlanet,PlayerNature,
                  Random(GameWorld.Size.X),Random(GameWorld.Size.Y));
    CreateNewUnit(FirePlanet,PlayerNature,
                  Random(GameWorld.Size.X),Random(GameWorld.Size.Y));
    CreateNewUnit(FirePlanet,PlayerNature,
                  Random(GameWorld.Size.X),Random(GameWorld.Size.Y));
    CreateNewUnit(FirePlanet,PlayerNature,
                  Random(GameWorld.Size.X),Random(GameWorld.Size.Y));
    For Z:=1 to 3 do
      Begin
        CreateNewUnit(Carrier          ,Player1,Random(1000),Random(1000));
        CreateNewUnit(BattleCruiseAmon ,Player1,Random(1000),Random(1000));
        CreateNewUnit(BattleCruiseBaton,Player1,Random(1000),Random(1000));
      End;
    For Z:=1 to 100 do
      CreateNewUnit(BattleCruiseBaton,Player2,14000+Z*40,14000+Z*40);
    WorldViewAt(100,100);
  End;

PROCEDURE TGame.WorldMiniMapUpdate;
  Var X,Y : Integer;
      Z   : TUnitCount;
  Begin
    ImageLock(MiniMapImage);
    ImageClear(MiniMapImage,0);
    For Z:=Low(TUnitCount) to
           High(TUnitCount) do
      With GameUnits[Z] do
        If (HitPoint>0) and
           (Owner<>NonePlayer) then
          Begin
            X:=Round(Pos.X) div GameWorld.Scale.X;
            Y:=Round(Pos.Y) div GameWorld.Scale.Y;
            ImageFillRect(MiniMapImage,X-1,Y-1,X+1,Y+1,PlayerColor[Owner]);
          End;
    X:=GameWorld.ViewPos.X div GameWorld.Scale.X;
    Y:=GameWorld.ViewPos.Y div GameWorld.Scale.Y;
    ImageRect(MiniMapImage,X,Y,
              X+ScrWidth div GameWorld.Scale.X,
              Y+ScrHeight div GameWorld.Scale.Y,White);
    ImageUnLock(MiniMapImage);
  End;

PROCEDURE TGame.ProceedUnitSettingFile(FileName : String);
  Var St,Sub : String;
```

```
    Typer  : TSpaceShip;
    F      : Text;
    Z      : Integer;
Procedure LoadAni(Var Data : TFrame);
  Var H      : THeading;
      Z      : Integer;
  Begin
    //Skip #Move#Attack#Somthing...
    ReadLn(F,St);
    For H:=H01 to H32 do
      Begin
        //Skip line [H#]
        ReadLn(F,St);
        ReadLn(F,Data[H].NumFrames);
        SetLength(Data[H].Ani,Data[H].NumFrames);
        For Z:=1 to Data[H].NumFrames do
          ReadLn(F,Data[H].Ani[Z-1].FramePos,
                   Data[H].Ani[Z-1].FrameStyle);
      End;
  End;
Begin
  Assign(F,FileName);
  Reset(F);
  ReadLn(F,St);
  St:=UpperCase(St);
  Sub:=GetSubString(St);
  If Sub=SettingStr then
    Begin
      Typer:=GetShipTyper(St);
      With UnitsProperty[Typer] do
        Begin
          _DrawLevel:=2;
          _ShipCanCreate:=Low(TSpaceShip);
          While Not EoF(F) do
            Begin
              ReadLn(F,St);
              //Ship comment or blank line
              If (St='') or (St[1]='#') then Continue;
              St:=UpperCase(St);
              Sub:=GetSubString(St);
              //Setting for image file
              If Sub=ImageFilesStr then
                Begin
                  Sub:=GetSubString(St);
                  _NumOfImages:=ToInt(Sub);
                  SetLength(_Images,_NumOfImages);
                  For Z:=1 to _NumOfImages do
                    Begin
                      ReadLn(F,St);
                      _Images[Z-1]:=LoadImage(DataPath+St,FormatA1R5G5B5);
                      SetAlphaMask(_Images[Z-1],0);
                    End;
                End
              Else
              //Ship size
              If Sub=ShipSizeStr then
                Begin
                  Sub:=GetSubString(St);
                  _Size:=ToInt(Sub);
                  _Size2:=_Size div 2;
                End
              Else
              //Ship hitpoint
              If Sub=ShipHitPointStr then
```

```
  Begin
    Sub:=GetSubString(St);
    _HitPoint:=ToInt(Sub);
  End
Else
//Ship damage
If Sub=ShipDamageStr then
  Begin
    Sub:=GetSubString(St);
    _Damage:=ToInt(Sub);
  End
Else
//Ship attack range
If Sub=ShipAttackRangeStr then
  Begin
    Sub:=GetSubString(St);
    _AttackRange:=ToInt(Sub);
  End
Else
//Ship speed
If Sub=ShipSpeedStr then
  Begin
    Sub:=GetSubString(St);
    _Speed:=ToInt(Sub);
  End
Else
//Ship wait
If Sub=ShipWaitStr then
  Begin
    Sub:=GetSubString(St);
    _Wait:=ToInt(Sub);
  End
Else
//Ship turn wait
If Sub=ShipTurnWaitStr then
  Begin
    Sub:=GetSubString(St);
    _TurnWait:=ToInt(Sub);
  End
Else
//Ship gun
If Sub=ShipGunStr then
  Begin
    Sub:=GetSubString(St);
    _Gun:=GetGun(Sub);
  End
Else
//Ship explosion
If Sub=ExplosionSkinStr then
  Begin
    Sub:=GetSubString(St);
    _Explosion:=GetShipTyper(Sub);
  End
Else
If Sub=ShipFrameStr then
  Begin
    LoadAni(_ShipAni.Move);
    LoadAni(_ShipAni.Attack);
  End
Else
If Sub=ShipCanCreate then
  Begin
    Sub:=GetSubString(St);
    _ShipCanCreate:=GetShipTyper(Sub);
```

```
                End
              Else
              If Sub=ShipCreateTime then
                Begin
                  Sub:=GetSubString(St);
                  _BuildTime:=ToInt(Sub);
                End
              Else
              If Sub=ShipCost then
                Begin
                  Sub:=GetSubString(St);
                  _Cost:=ToInt(Sub);
                End;
            End;
        End;
      End;
    Close(F);
  End;

PROCEDURE TGame.SetupUnitsSetting;
  Begin
    UnitsProperty[FirePlanet]._DrawLevel:=1;
    ProceedUnitSettingFile(DataPath+ShipsSettingDir+'ShipCarrier.Config');

ProceedUnitSettingFile(DataPath+ShipsSettingDir+'ShipBattleCruiseAmon.Config')
;

ProceedUnitSettingFile(DataPath+ShipsSettingDir+'ShipBattleCruiseBaton.Config'
);
    ProceedUnitSettingFile(DataPath+ShipsSettingDir+'ShipCooNok.Config');
    ProceedUnitSettingFile(DataPath+ShipsSettingDir+'ShipScout.Config');
    ProceedUnitSettingFile(DataPath+ShipsSettingDir+'EvelMissile.Config');
    ProceedUnitSettingFile(DataPath+ShipsSettingDir+'MileMissile.Config');
    ProceedUnitSettingFile(DataPath+ShipsSettingDir+'MiniMissile.Config');
    ProceedUnitSettingFile(DataPath+ShipsSettingDir+'Burn.Config');
  End;

PROCEDURE TGame.SetupUnitsImage;
  Begin
    GlobeImageNum:=LoadImage(DataPath+'Globe.jpg',FormatA1R5G5B5);
    FlareImageNum:=LoadImage(DataPath+'Flare.jpg',FormatA1R5G5B5);
    SetAlphaMask(GlobeImageNum,0);
  End;

FUNCTION  TGame.CreateNewUnit(UnitTyper : TSpaceShip;
                              Player : TPlayer;
                              PosX,PosY : Integer) : TUnitCount;
  Var Z : TUnitCount;
  Begin
    For Z:=Low(TUnitCount)+1 to
           High(TUnitCount) do
      If GameUnits[Z].HitPoint<=0 then
        Begin
          GameUnits[Z].Pos.X:=PosX;
          GameUnits[Z].Pos.Y:=PosY;
          SetupUnit(Z,UnitTyper,Player);
          Result:=Z;
          Inc(GamePlayers[Player].CountUnit);
          Inc(UnitCounting);
          Exit;
        End;
    Result:=Low(TUnitCount);
  End;
```

```
FUNCTION  TGame.DestroyUnit(UnitNum : TUnitCount) : Boolean;
  Var Z : TUnitCount;
  Begin
    Result:=True;
    With GameUnits[UnitNum] do
      Begin
        If HitPoint<=0 then Exit;
        If FirstUnitSelect=UnitNum then
          FirstUnitSelect:=Low(TUnitCount);
        For Z:=Low(TUnitCount) to
               High(TUnitCount) do
          If (GameUnits[Z].HitPoint>0) and
             (GameUnits[Z].UnitTarget=UnitNum) then
            Begin
              If (GameUnits[Z].Owner<>NonePlayer) then
                UnitResetCommand(Z)
              Else DestroyUnit(Z);
            End;
        Dec(GamePlayers[Owner].CountUnit);
        Dec(UnitCounting);
        HitPoint:=0;
      End;
  End;

PROCEDURE TGame.SetupUnit(UnitNum : TUnitCount;
                          UnitTyper : TSpaceShip;
                          Player : TPlayer);
  Begin
    With GameUnits[UnitNum] do
      Begin
        Owner:=Player;
        Typer:=UnitTyper;
        HitPoint:=UnitsProperty[UnitTyper]._HitPoint;
        UnitCmd:=NoCmd;
        UnitNextCmd:=NoCmd;
        CurrentFrame:=0;
        CreationCounting:=0;
        Case Typer of
          FirePlanet :
            Begin
              TransparentLevel:=Random(255);
              TransparentSpeed:=1;
            End;
          Else
            Begin
              CurrentHead:=RandomHeading;
              UnitGroup:=0;
            End;
        End;
      End;
  End;

FUNCTION  TGame.CostForNewUnit(Player : TPlayer;
                               UnitTyper : TSpaceShip) : Boolean;
  Begin
    With GamePlayers[Player] do
      Begin
        If Money>=UnitsProperty[UnitTyper]._Cost then
          Begin
            Dec(Money,UnitsProperty[UnitTyper]._Cost);
            Result:=True;
          End
        Else Result:=False;
      End;
```

```
  End;

PROCEDURE TGame.SelectUnit(UnitNum : TUnitCount);
  Begin
    With GameUnits[UnitNum] do
      Begin
        If FirstUnitSelect=Low(TUnitCount) then
          FirstUnitSelect:=UnitNum;
        If UnitGroup and 128=0 then
          UnitGroup:=UnitGroup xor 128;
      End;
  End;

PROCEDURE TGame.UnSelectAllUnit;
  Var Z : TUnitCount;
  Begin
    FirstUnitSelect:=Low(TUnitCount);
    For Z:=Low(TUnitCount) to
           High(TUnitCount) do
      With GameUnits[Z] do
        UnitGroup:=UnitGroup and 127;
  End;

FUNCTION  TGame.GetUnitAtMouse(MX,MY : Integer;
                               HumanOwner : Boolean) : TUnitCount;
  Var Ok      : Boolean;
      Z,Saved : TUnitCount;
      XX,YY   : Single;
      SprNum  : Integer;
  Begin
    Saved:=Low(TUnitCount);
    For Z:=Low(TUnitCount) to
           High(TUnitCount) do
      With GameUnits[Z],
           UnitsProperty[Typer] do
        If (HitPoint>0) and
           (Owner<>NonePlayer) then
          Begin
            If HumanOwner and
               (Owner<>HumanControl) then Continue;
            XX:=Pos.X-GameWorld.ViewPos.X;
            YY:=Pos.Y-GameWorld.ViewPos.Y;
            Ok:=(MX>=XX-_Size2) and
                (MX<=XX+_Size2) and
                (MY>=YY-_Size2) and
                (MY<=YY+_Size2);
            If Ok then
              Begin
                Case UnitCmd of
                  NoCmd,CmdMove :
                    Begin
                      SprNum:=_ShipAni.Move[CurrentHead].Ani[0].FramePos;
                      XX:=MX+GameWorld.ViewPos.X-Pos.X+
                          GetImageWidth(_Images[SprNum]) ShR 1;
                      YY:=MY+GameWorld.ViewPos.Y-Pos.Y+
                          GetImageHeight(_Images[SprNum]) ShR 1;
                      Ok:=ImageGetPixel(_Images[SprNum],
                                        Round(XX),Round(YY))<>0;
                    End;
                End;
                If Ok then Saved:=Z;
              End;
          End;
    Result:=Saved;
```

```
  End;

PROCEDURE TGame.CallMouseSelectionUnit(Add : Boolean);
  Var Saved,Z     : TUnitCount;
      X1,Y1,X2,Y2 : Single;
  Begin
    If (SelectStart.X=SelectEnd.X) and
       (SelectStart.Y=SelectEnd.Y) then
      Begin
        Saved:=GetUnitAtMouse(SelectStart.X,SelectStart.Y,True);
        If Saved<>Low(TUnitCount) then
          Begin
            If Not Add then
              UnSelectAllUnit;
            SelectUnit(Saved);
          End;
      End
    Else
      Begin
        If Not Add then
          UnSelectAllUnit;
        X1:=SelectStart.X+GameWorld.ViewPos.X;
        Y1:=SelectStart.Y+GameWorld.ViewPos.Y;
        X2:=SelectEnd.X+GameWorld.ViewPos.X;
        Y2:=SelectEnd.Y+GameWorld.ViewPos.Y;
        For Z:=Low(TUnitCount) to
               High(TUnitCount) do
          With GameUnits[Z],
               UnitsProperty[Typer] do
            If (HitPoint>0) and
               (Owner=HumanControl) then
              Begin
                If (X1<=Pos.X+_Size2) and
                   (X2>=Pos.X-_Size2) and
                   (Y1<=Pos.Y+_Size2) and
                   (Y2>=Pos.Y-_Size2) then SelectUnit(Z);
              End;
      End;
  End;

FUNCTION  TGame.UnitGetBestHeading(UnitNum : TUnitCount) : THeading;
  Var H,Best       : THeading;
      Len,Min,X,Y  : Single;
  Begin
    Min:=High(Integer);
    Best:=Low(THeading);
    With GameUnits[UnitNum] do
      For H:=H01 to H32 do
        Begin
          X:=Pos.X+Moving[H,1];
          Y:=Pos.Y+Moving[H,2];
          Len:=Sqr(X-Dest.X)+Sqr(Y-Dest.Y);
          If Len<Min then
            Begin
              Min:=Len;
              Best:=H;
            End;
        End;
    Result:=Best;
  End;

FUNCTION  TGame.UnitCloseToTarget(UnitNum : TUnitCount;
                                  Long : Integer) : Boolean;
  Var Len : Single;
```

```
  Begin
    With GameUnits[UnitNum] do
      Begin
        Len:=Sqr(Pos.X-Dest.X)+Sqr(Pos.Y-Dest.Y);
        Result:=Sqrt(Len)<=Long;
      End;
  End;

PROCEDURE TGame.UnitPointToTarget(UnitNum : TUnitCount);
  Begin
    With GameUnits[UnitNum] do
      Begin
        Head:=UnitGetBestHeading(UnitNum);
      End;
  End;

FUNCTION  TGame.UnitVisible(UnitNum : TUnitCount) : Boolean;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Result:=(Pos.X-_Size2<GameWorld.ViewPos.X+ScrWidth) and
              (Pos.Y-_Size2<GameWorld.ViewPos.Y+ScrHeight) and
              (Pos.X+_Size2>GameWorld.ViewPos.X) and
              (Pos.Y+_Size2>GameWorld.ViewPos.Y);
  End;

FUNCTION  TGame.HeadNear(StartHead,EndHead : THeading) : THeading;
  Var L,R : Integer;
  Begin
    If StartHead>EndHead then
      Begin
        L:=Integer(StartHead)-Integer(EndHead);
        R:=Integer(High(THeading))-Integer(StartHead)+
           Integer(EndHead)-Integer(Low(THeading));
      End
    Else
      Begin
        R:=Integer(EndHead)-Integer(StartHead);
        L:=Integer(High(THeading))-Integer(EndHead)+
           Integer(StartHead)-Integer(Low(THeading));
      End;
    If L<R then
      Begin
        If StartHead>Low(THeading) then Dec(StartHead)
        Else StartHead:=High(THeading);
      End
    Else
      Begin
        If StartHead<High(THeading) then Inc(StartHead)
        Else StartHead:=Low(THeading);
      End;
    Result:=StartHead;
  End;

PROCEDURE TGame.PointNear(X,Y,Leng : Integer;Var DX,DY : Integer);
  Var I,J : Integer;
  Begin
    I:=Random(Leng);
    J:=Random(Leng);
    Case Random(2) of
      0 : DX:=X+I;
      1 : DX:=X-I;
    End;
    Case Random(2) of
```

```
      0 : DY:=Y+J;
      1 : DY:=Y-J;
    End;
  End;

FUNCTION  TGame.Long(X1,Y1,X2,Y2 : Single) : Single;
  Begin
    Result:=Sqr(X1-X2)+Sqr(Y1-Y2);
  End;

FUNCTION  TGame.RandomPlayer : TPlayer;
  Begin
    Case Random(4) of
      0 : Result:=Player1;
      1 : Result:=Player2;
      2 : Result:=Player3;
      3 : Result:=Player4;
      Else Result:=NonePlayer;
    End;
  End;

PROCEDURE TGame.UnitFire(UnitNum : TUnitCount);
  Var Num : TUnitCount;
      X,Y : Integer;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        Case _Gun of
          GunMiniMissile :
            Begin
              Num:=CreateNewUnit(MiniMissile,NonePlayer,
                                 Round(Pos.X),Round(Pos.Y));
              If Num<>Low(TUnitCount) then
                Begin
                  //Get a random point nearby target !
                  PointNear(Round(GameUnits[UnitTarget].Pos.X),
                            Round(GameUnits[UnitTarget].Pos.Y),
                            Min(_Size2 ShR 1,_Size2 ShR 1),X,Y);
                  GameUnits[Num].Dest.Y:=Y;
                  GameUnits[Num].Dest.X:=X;
                  GameUnits[Num].UnitCmd:=CmdFire;
                  GameUnits[Num].Head:=UnitGetBestHeading(Num);
                  GameUnits[Num].CurrentHead:=GameUnits[Num].Head;
                  GameUnits[Num].UnitTarget:=UnitTarget;
                  //
                  //Play missile lauched
                  //
                  {$IfDef SoundPlaying}
                  If UnitVisible(Num) then
                    If LastMissileLauchSoundTime<GameTime-2000 then
                      Begin
                        LastMissileLauchSoundTime:=GameTime;
                        SoundPlay(MissileLauch);
                      End;
                  {$EndIf}
                End
              Else
                Begin
                  CreateMessage('Unit limit exceed !');
                End;
            End;
          GunEvelMissile :
            Begin
```

```
              Num:=CreateNewUnit(EvelMissile,NonePlayer,
                                 Round(Pos.X),Round(Pos.Y));
              If Num<>Low(TUnitCount) then
                Begin
                  //Get a random point nearby target !
                  PointNear(Round(GameUnits[UnitTarget].Pos.X),
                            Round(GameUnits[UnitTarget].Pos.Y),
                            Min(_Size2 ShR 1,_Size2 ShR 1),X,Y);
                  GameUnits[Num].Dest.Y:=Y;
                  GameUnits[Num].Dest.X:=X;
                  GameUnits[Num].UnitCmd:=CmdFire;
                  GameUnits[Num].Head:=UnitGetBestHeading(Num);
                  GameUnits[Num].CurrentHead:=GameUnits[Num].Head;
                  GameUnits[Num].UnitTarget:=UnitTarget;
                  //
                  //Play missile lauched
                  //
                  {$IfDef SoundPlaying}
                  If UnitVisible(Num) then
                    If LastMissileLauchSoundTime<GameTime-2000 then
                      Begin
                        LastMissileLauchSoundTime:=GameTime;
                        SoundPlay(MissileLauch);
                      End;
                  {$EndIf}
                End
              Else
                Begin
                  CreateMessage('Unit limit exceed !');
                End;
            End;
          GunMileMissile :
            Begin
              Num:=CreateNewUnit(MileMissile,NonePlayer,
                                 Round(Pos.X),Round(Pos.Y));
              If Num<>Low(TUnitCount) then
                Begin
                  //Get a random point nearby target !
                  PointNear(Round(GameUnits[UnitTarget].Pos.X),
                            Round(GameUnits[UnitTarget].Pos.Y),
                            Min(_Size2 ShR 1,_Size2 ShR 1),X,Y);
                  GameUnits[Num].Dest.Y:=Y;
                  GameUnits[Num].Dest.X:=X;
                  GameUnits[Num].UnitCmd:=CmdFire;
                  GameUnits[Num].Head:=UnitGetBestHeading(Num);
                  GameUnits[Num].CurrentHead:=GameUnits[Num].Head;
                  GameUnits[Num].UnitTarget:=UnitTarget;
                End
              Else
                Begin
                  CreateMessage('Unit limit exceed !');
                End;
            End;
        End;
    End;
  End;

PROCEDURE TGame.UnitHit(UnitNum : TUnitCount);
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        UnitDecHitPoint(UnitTarget,_Damage);
      End;
```

```
  End;

PROCEDURE TGame.UnitDecHitPoint(UnitNum : TUnitCount;Damage : Integer);
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        If HitPoint>Damage then Dec(HitPoint,Damage)
        Else DestroyUnit(UnitNum);
      End;
  End;

FUNCTION  TGame.UnitGetNextCmd(UnitNum : TUnitCount) : Boolean;
  Begin
    With GameUnits[UnitNum] do
      Begin
        If UnitNextCmd<>NoCmd then
          Begin
            UnitCmd:=UnitNextCmd;
            UnitTarget:=UnitNextTarget;
            UnitNextCmd:=NoCmd;
            Case UnitCmd of
              CmdMove,CmdPatrol :
                Begin
                  UnitPointToTarget(UnitNum);
                End;
              CmdAttack,CmdAttackP,CmdAttackM :
                Begin
                  Dest:=GameUnits[UnitTarget].Pos;
                  UnitPointToTarget(UnitNum);
                End;
              CmdAttackT :
                Begin
                  UnitPointToTarget(UnitNum);
                End;
            End;
            Result:=True;
          End
        Else Result:=False;
      End;
  End;

PROCEDURE TGame.UnitResetCommand(UnitNum : TUnitCount);
  Begin
    With GameUnits[UnitNum] do
      Begin
        If UnitCmd=CmdAttackP then
          Begin
            UnitCmd:=CmdPatrol;
            Dest:=PatrolDest;
            UnitPointToTarget(UnitNum);
          End
        Else
        If UnitCmd=CmdAttackM then
          Begin
            UnitCmd:=CmdAttackT;
            Dest:=PatrolDest;
            UnitPointToTarget(UnitNum);
          End
        Else UnitCmd:=NoCmd;
        UnitTarget:=Low(TUnitCount);
      End;
  End;
```

```
PROCEDURE TGame.UnitSetCreate(UnitNum : TUnitCount);
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        If _ShipCanCreate=Low(TSpaceShip) then Exit;
        If CreationCounting>0 then Exit;
        If CostForNewUnit(Owner,_ShipCanCreate) then
          CreationCounting:=1;
      End;
  End;

PROCEDURE TGame.UnitSetMove(UnitNum : TUnitCount;DestX,DestY : Integer);
  Begin
    With GameUnits[UnitNum] do
      Begin
        If UnitCmd=NoCmd then
          Begin
            UnitCmd:=CmdMove;
            UnitTarget:=Low(TUnitCount);
            Dest.X:=DestX;
            Dest.Y:=DestY;
            UnitPointToTarget(UnitNum);
          End
        Else
          Begin
            UnitNextCmd:=CmdMove;
            UnitNextTarget:=Low(TUnitCount);
            Dest.X:=DestX;
            Dest.Y:=DestY;
          End;
      End;
  End;

PROCEDURE TGame.UnitSetPatrol(UnitNum : TUnitCount;DestX,DestY : Integer);
  Begin
    With GameUnits[UnitNum] do
      Begin
        If (UnitCmd=NoCmd) or
           (UnitCmd=CmdPatrol) then
          Begin
            UnitCmd:=CmdPatrol;
            UnitTarget:=Low(TUnitCount);
            PatrolStart:=Pos;
            PatrolDest.X:=DestX;
            PatrolDest.Y:=DestY;
            Dest:=PatrolDest;
            UnitPointToTarget(UnitNum);
          End
        Else
          Begin
            UnitNextCmd:=CmdPatrol;
            UnitNextTarget:=Low(TUnitCount);
            PatrolStart:=Pos;
            PatrolDest.X:=DestX;
            PatrolDest.Y:=DestY;
          End;
      End;
  End;

PROCEDURE TGame.UnitSetAttackTo(UnitNum : TUnitCount;DestX,DestY : Integer);
  Begin
    With GameUnits[UnitNum] do
      Begin
```

```
        If (UnitCmd=NoCmd) or
           (UnitCmd=CmdPatrol) then
          Begin
            UnitCmd:=CmdAttackT;
            UnitTarget:=Low(TUnitCount);
            Dest.X:=DestX;
            Dest.Y:=DestY;
            UnitPointToTarget(UnitNum);
          End
        Else
          Begin
            UnitNextCmd:=CmdAttackT;
            UnitNextTarget:=Low(TUnitCount);
            Dest.X:=DestX;
            Dest.Y:=DestY;
          End;
      End;
  End;

PROCEDURE TGame.UnitSetAttack(UnitNum : TUnitCount;Target : Integer);
  Begin
    If Target=UnitNum then Exit;
    With GameUnits[UnitNum] do
      Begin
        If UnitCmd=NoCmd then
          Begin
            UnitCmd:=CmdAttack;
            UnitTarget:=Target;
            Dest:=GameUnits[Target].Pos;
            UnitPointToTarget(UnitNum);
          End
        Else
          Begin
            UnitNextCmd:=CmdAttack;
            UnitNextTarget:=Target;
          End;
      End;
  End;

PROCEDURE TGame.UnitSetAttackPatrol(UnitNum : TUnitCount;Target : Integer);
  Begin
    If Target=UnitNum then Exit;
    With GameUnits[UnitNum] do
      Begin
        If UnitCmd=NoCmd then
          Begin
            UnitCmd:=CmdAttackP;
            UnitTarget:=Target;
            Dest:=GameUnits[Target].Pos;
            UnitPointToTarget(UnitNum);
          End
        Else
          Begin
            UnitNextCmd:=CmdAttackP;
            UnitNextTarget:=Target;
          End;
      End;
  End;

PROCEDURE TGame.UnitSetAttackMove(UnitNum : TUnitCount;Target : Integer);
  Begin
    If Target=UnitNum then Exit;
    With GameUnits[UnitNum] do
      Begin
```

```
        If UnitCmd=NoCmd then
          Begin
            UnitCmd:=CmdAttackM;
            UnitTarget:=Target;
            PatrolDest:=Dest;
            Dest:=GameUnits[Target].Pos;
            UnitPointToTarget(UnitNum);
          End
        Else
          Begin
            PatrolDest:=Dest;
            UnitNextCmd:=CmdAttackM;
            UnitNextTarget:=Target;
          End;
      End;
  End;

FUNCTION  TGame.UnitFindTarget(UnitNum : TUnitCount) : TUnitCount;
  Var Z,Saved  : TUnitCount;
      Min,Leng : Single;
  Begin
    Min:=High(Integer);
    Saved:=Low(TUnitCount);
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      For Z:=Low(TUnitCount) to
             High(TUnitCount) do
        If (GamePlayers[Owner].Enemy[GameUnits[Z].Owner]) and
           (GameUnits[Z].HitPoint>0) then
          Begin
            Leng:=Long(Pos.X,Pos.Y,
                       GameUnits[Z].Pos.X,
                       GameUnits[Z].Pos.Y);
            If (Leng<Min) and
               (Leng<Sqr(_AttackRange)) then
              Begin
                Min:=Leng;
                Saved:=Z;
              End;
          End;
    Result:=Saved;
  End;

FUNCTION  TGame.UnitFindNear(UnitNum : TUnitCount) : TUnitCount;
  Var Z,Saved  : TUnitCount;
      Min,Leng : Single;
  Begin
    Min:=High(Integer);
    Saved:=Low(TUnitCount);
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      For Z:=Low(TUnitCount) to
             High(TUnitCount) do
        If (Z<>UnitNum) and
           (GameUnits[Z].Owner=Owner) and
           (GameUnits[Z].HitPoint>0) then
          Begin
            Leng:=Long(Pos.X,Pos.Y,
                       GameUnits[Z].Pos.X,
                       GameUnits[Z].Pos.Y);
            If (Leng<Min) and
               (Leng<Sqr(TooNearLeng)) then
              Begin
                Min:=Leng;
```

```
              Saved:=Z;
            End;
        End;
    Result:=Saved;
  End;

PROCEDURE TGame.ProcessSpecifyUnit(UnitNum : TUnitCount);
  Begin
    With GameUnits[UnitNum] do
      Begin
      End;
  End;

PROCEDURE TGame.ProcessNormalUnit(UnitNum : TUnitCount);
  Var Num : TUnitCount;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        //Unit is owner by computer ?
        If Owner<>HumanControl then
          Begin
            //Can be create ?
            If (_ShipCanCreate<>Low(TSpaceShip)) and
            //Now not create ?
               (CreationCounting=0) then
              Begin
                //Create unit !
                UnitSetCreate(UnitNum);
              End;
          End;
        //Unit creation process
        If CreationCounting>0 then
          Begin
            Inc(CreationCounting);
            If CreationCounting>UnitsProperty[_ShipCanCreate]._BuildTime then
              Begin
                CreationCounting:=0;
                Num:=CreateNewUnit(_ShipCanCreate,Owner,
                                   Round(Pos.X),Round(Pos.Y));
                UnitSetMove(Num,Round(Pos.X)-150+Random(300),
                                Round(Pos.Y)-150+Random(300));
              End;
          End;
        //Unit command process
        Case UnitCmd of
          CmdMove,CmdFire : ProceedUnitMove(UnitNum);
          CmdAttack       : ProceedUnitAttack(UnitNum);
          CmdAttackP      : ProceedUnitAttackP(UnitNum);
          CmdAttackT      : ProceedUnitAttackT(UnitNum);
          CmdAttackM      : ProceedUnitAttackM(UnitNum);
          CmdPatrol       : ProceedUnitPatrol(UnitNum);
          CmdBurn         : ProceedUnitBurning(UnitNum);
          Else ProceedUnitStand(UnitNum);
        End;
      End;
  End;

PROCEDURE TGame.ProceedUnitStand(UnitNum : TUnitCount);
  Var Num : TUnitCount;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
```

```
        If IsFrameToFindTarget then
          Begin
            Num:=UnitFindTarget(UnitNum);
            If Num<>Low(TUnitCount) then
              Begin
                UnitSetAttack(UnitNum,Num);
                Exit;
              End
            Else
              Begin
              End;
          End;
        If WaitTimeForMove>0 then
          Begin
            Dec(WaitTimeForMove);
            Exit;
          End;
        If UnitGetNextCmd(UnitNum) then Exit;
        //Unit is owner by computer ?
        If Owner<>HumanControl then
          Begin
            UnitSetPatrol(UnitNum,Random(GameWorld.Size.X),
                                  Random(GameWorld.Size.Y));
          End;
        WaitTimeForMove:=_TurnWait;
      End;
  End;

PROCEDURE TGame.ProceedUnitMove(UnitNum : TUnitCount);
  Var _Heading : THeading;
      Num      : TUnitCount;
      Z        : Integer;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        If WaitTimeForMove>0 then
          Begin
            Dec(WaitTimeForMove);
            Exit;
          End;
        If UnitGetNextCmd(UnitNum) then Exit;
        If UnitCloseToTarget(UnitNum,MoveFarLeng) then
          Begin
            If UnitCmd=CmdFire then
              Begin
                {$IfDef SoundPlaying}
                If UnitVisible(UnitNum) then
                  If LastMissileExploredSoundTime<GameTime-2000 then
                    Begin
                      LastMissileExploredSoundTime:=GameTime;
                      SoundPlay(MissileExplored);
                    End;
                {$EndIf}
                UnitHit(UnitNum);
                DestroyUnit(UnitNum);
                Exit;
              End
            Else
            If UnitGetNextCmd(UnitNum) then
            Else UnitCmd:=NoCmd;
          End
        Else
        //Turn ship ?
```

```
        If CurrentHead<>Head then
          Begin
            CurrentHead:=HeadNear(CurrentHead,Head);
            WaitTimeForMove:=_TurnWait;
          End
        Else
          Begin
            For Z:=1 to _Speed do
              Begin
                _Heading:=UnitGetBestHeading(UnitNum);
                Pos.X:=Pos.X+Moving[_Heading,1];
                Pos.Y:=Pos.Y+Moving[_Heading,2];
              End;
            WaitTimeForMove:=_Wait;
            //Increase unit frame
            If CurrentFrame<
               _ShipAni.Move[CurrentHead].NumFrames-1 then Inc(CurrentFrame)
            Else CurrentFrame:=0;
            //Missile generating the burning
            If UnitCmd=CmdFire then
              Begin
                Num:=CreateNewUnit(Burn,NonePlayer,
                                   Round(Pos.X),Round(Pos.Y));
                If Num<>Low(TUnitCount) then
                  Begin
                    //Unit is burning !
                    GameUnits[Num].UnitCmd:=CmdBurn;
                  End;
              End;
          End;
      End;
  End;

PROCEDURE TGame.ProceedUnitAttack(UnitNum : TUnitCount);
  Var _Heading : THeading;
      Z        : Integer;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        If WaitTimeForMove>0 then
          Begin
            Dec(WaitTimeForMove);
            Exit;
          End;
        If UnitGetNextCmd(UnitNum) then Exit;
        If GameUnits[UnitTarget].HitPoint<=0 then
          Begin
            UnitCmd:=NoCmd;
            Exit;
          End
        Else
          Begin
            If (UnitTarget<>0) and
               ((Dest.X<>GameUnits[UnitTarget].Pos.X) or
                (Dest.Y<>GameUnits[UnitTarget].Pos.Y))then
              Begin
                Dest:=GameUnits[UnitTarget].Pos;
                Head:=UnitGetBestHeading(UnitNum);
              End;
          End;
        If UnitCloseToTarget(UnitNum,_AttackRange) then
          Begin
            UnitFire(UnitNum);
```

```
            WaitTimeForMove:=_TurnWait*20;
          End
        Else
        //Turn ship ?
        If CurrentHead<>Head then
          Begin
            CurrentHead:=HeadNear(CurrentHead,Head);
            WaitTimeForMove:=_TurnWait;
          End
        Else
          Begin
            For Z:=1 to _Speed do
              Begin
                _Heading:=UnitGetBestHeading(UnitNum);
                Pos.X:=Pos.X+Moving[_Heading,1];
                Pos.Y:=Pos.Y+Moving[_Heading,2];
              End;
            WaitTimeForMove:=_Wait;
          End;
      End;
  End;

PROCEDURE TGame.ProceedUnitAttackP(UnitNum : TUnitCount);
  Var _Heading : THeading;
      Z        : Integer;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        If WaitTimeForMove>0 then
          Begin
            Dec(WaitTimeForMove);
            Exit;
          End;
        If UnitGetNextCmd(UnitNum) then Exit;
        If GameUnits[UnitTarget].HitPoint<=0 then
          Begin
            //Back to patrol command
            UnitCmd:=CmdPatrol;
            Dest:=PatrolDest;
            UnitPointToTarget(UnitNum);
            Exit;
          End
        Else
          Begin
            If (UnitTarget<>0) and
               ((Dest.X<>GameUnits[UnitTarget].Pos.X) or
                (Dest.Y<>GameUnits[UnitTarget].Pos.Y))then
              Begin
                Dest:=GameUnits[UnitTarget].Pos;
                Head:=UnitGetBestHeading(UnitNum);
              End;
          End;
        If UnitCloseToTarget(UnitNum,_AttackRange) then
          Begin
            UnitFire(UnitNum);
            WaitTimeForMove:=_TurnWait*20;
          End
        Else
        //Turn ship ?
        If CurrentHead<>Head then
          Begin
            CurrentHead:=HeadNear(CurrentHead,Head);
            WaitTimeForMove:=_TurnWait;
```

```
          End
        Else
          Begin
            For Z:=1 to _Speed do
              Begin
                _Heading:=UnitGetBestHeading(UnitNum);
                Pos.X:=Pos.X+Moving[_Heading,1];
                Pos.Y:=Pos.Y+Moving[_Heading,2];
              End;
            WaitTimeForMove:=_Wait;
          End;
      End;
  End;

PROCEDURE TGame.ProceedUnitAttackM(UnitNum : TUnitCount);
  Var _Heading : THeading;
      Z        : Integer;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        If WaitTimeForMove>0 then
          Begin
            Dec(WaitTimeForMove);
            Exit;
          End;
        If UnitGetNextCmd(UnitNum) then Exit;
        If GameUnits[UnitTarget].HitPoint<=0 then
          Begin
            //Back to patrol command
            UnitCmd:=CmdAttackT;
            Dest:=PatrolDest;
            UnitPointToTarget(UnitNum);
            Exit;
          End
        Else
          Begin
            If (UnitTarget<>0) and
               ((Dest.X<>GameUnits[UnitTarget].Pos.X) or
                (Dest.Y<>GameUnits[UnitTarget].Pos.Y))then
              Begin
                Dest:=GameUnits[UnitTarget].Pos;
                Head:=UnitGetBestHeading(UnitNum);
              End;
          End;
        If UnitCloseToTarget(UnitNum,_AttackRange) then
          Begin
            UnitFire(UnitNum);
            WaitTimeForMove:=_TurnWait*20;
          End
        Else
        //Turn ship ?
        If CurrentHead<>Head then
          Begin
            CurrentHead:=HeadNear(CurrentHead,Head);
            WaitTimeForMove:=_TurnWait;
          End
        Else
          Begin
            For Z:=1 to _Speed do
              Begin
                _Heading:=UnitGetBestHeading(UnitNum);
                Pos.X:=Pos.X+Moving[_Heading,1];
                Pos.Y:=Pos.Y+Moving[_Heading,2];
```

```
            End;
          WaitTimeForMove:=_Wait;
        End;
    End;
  End;

PROCEDURE TGame.ProceedUnitAttackT(UnitNum : TUnitCount);
  Var _Heading : THeading;
      Num      : TUnitCount;
      Z        : Integer;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        If WaitTimeForMove>0 then
          Begin
            Dec(WaitTimeForMove);
            Exit;
          End;
        If UnitGetNextCmd(UnitNum) then Exit;
        If UnitCloseToTarget(UnitNum,MoveFarLeng) then
          Begin
            If UnitGetNextCmd(UnitNum) then
            Else UnitCmd:=NoCmd;
          End
        Else
        //Turn ship ?
        If CurrentHead<>Head then
          Begin
            CurrentHead:=HeadNear(CurrentHead,Head);
            WaitTimeForMove:=_TurnWait;
          End
        Else
          Begin
            For Z:=1 to _Speed do
              Begin
                _Heading:=UnitGetBestHeading(UnitNum);
                Pos.X:=Pos.X+Moving[_Heading,1];
                Pos.Y:=Pos.Y+Moving[_Heading,2];
              End;
            If IsFrameToFindTarget then
              Begin
                Num:=UnitFindTarget(UnitNum);
                If Num<>Low(TUnitCount) then
                  Begin
                    UnitCmd:=NoCmd;
                    UnitSetAttackMove(UnitNum,Num);
                    Exit;
                  End;
              End;
            WaitTimeForMove:=_Wait;
            //Increase unit frame
            If CurrentFrame<
               _ShipAni.Move[CurrentHead].NumFrames-1 then Inc(CurrentFrame)
            Else CurrentFrame:=0;
          End;
      End;
  End;

PROCEDURE TGame.ProceedUnitPatrol(UnitNum : TUnitCount);
  Var _Heading : THeading;
      Z        : Integer;
      Num      : TUnitCount;
  Begin
```

```
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        If WaitTimeForMove>0 then
          Begin
            Dec(WaitTimeForMove);
            Exit;
          End;
        If UnitGetNextCmd(UnitNum) then Exit;
        If UnitCloseToTarget(UnitNum,MoveFarLeng) then
          Begin
            If UnitGetNextCmd(UnitNum) then
            Else
              Begin
                Dest:=PatrolStart;
                PatrolStart:=PatrolDest;
                PatrolDest:=Dest;
                UnitPointToTarget(UnitNum);
              End;
          End
        Else
        //Turn ship ?
        If CurrentHead<>Head then
          Begin
            CurrentHead:=HeadNear(CurrentHead,Head);
            WaitTimeForMove:=_TurnWait;
          End
        Else
          Begin
            For Z:=1 to _Speed do
              Begin
                _Heading:=UnitGetBestHeading(UnitNum);
                Pos.X:=Pos.X+Moving[_Heading,1];
                Pos.Y:=Pos.Y+Moving[_Heading,2];
              End;
            If IsFrameToFindTarget then
              Begin
                Num:=UnitFindTarget(UnitNum);
                If Num<>Low(TUnitCount) then
                  Begin
                    UnitCmd:=NoCmd;
                    UnitSetAttackPatrol(UnitNum,Num);
                    Exit;
                  End;
              End;
            WaitTimeForMove:=_Wait;
            //Increase unit frame
            If CurrentFrame<
               _ShipAni.Move[CurrentHead].NumFrames-1 then Inc(CurrentFrame)
            Else CurrentFrame:=0;
          End;
      End;
  End;

PROCEDURE TGame.ProceedUnitBurning(UnitNum : TUnitCount);
  Begin
    //When unit burning, unit do nothing else burning !
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      Begin
        If WaitTimeForMove>0 then
          Begin
            Dec(WaitTimeForMove);
            Exit;
```

```
          End;
        WaitTimeForMove:=_Wait;
        //Increase unit frame
        If CurrentFrame<_ShipAni.Move[CurrentHead].NumFrames-1 then
          Begin
            Inc(CurrentFrame);
            Pos.X:=Pos.X-Random(3);
            Pos.Y:=Pos.Y-Random(3);
          End
        Else
          Begin
            DestroyUnit(UnitNum);
          End;
      End;
  End;

PROCEDURE TGame.RenderWorld;
  Var I,J,X,Y : Integer;
  Begin
    With GameWorld do
      Begin
        X:=ViewPos.X mod GetImageWidth(StarBackGround);
        Y:=ViewPos.Y mod GetImageHeight(StarBackGround);
        For I:=-1 to NumTile.X+1 do
          For J:=-1 to NumTile.Y+1 do
            ImageRenderEffect(I*GetImageWidth(StarBackGround)-X,
                              J*GetImageHeight(StarBackGround)-Y,
                              StarBackGround,EffectNone);
      End;
  End;

PROCEDURE TGame.RenderUnits;
  Var DrawLevel : Integer;
      Z         : TUnitCount;
  Begin
    For DrawLevel:=1 to 2 do
      For Z:=Low(TUnitCount) to
             High(TUnitCount) do
        With GameUnits[Z],
             UnitsProperty[Typer] do
          If (HitPoint>0) and
             (_DrawLevel=DrawLevel) then
            Begin
              RenderUnits(Z);
            End;
  End;

PROCEDURE TGame.RenderUnits(UnitNum : TUnitCount);
  Var X,Y,X1,Y1,X2,Y2,
      SprNum,Effects : Integer;
      Visible        : Boolean;
  Begin
    With GameUnits[UnitNum],
         UnitsProperty[Typer] do
      If HitPoint>0 then
        Begin
          //Unit visible ?
          Visible:=UnitVisible(UnitNum);
          //Drawing unit
          X:=Round(Pos.X)-GameWorld.ViewPos.X;
          Y:=Round(Pos.Y)-GameWorld.ViewPos.Y;
          If Visible then
            Case Typer of
              FirePlanet :
```

```
                Begin
                  ImageRenderEffect(X-GetImageWidth(GlobeImageNum) ShR 1,
                                    Y-GetImageHeight(GlobeImageNum) ShR 1,
                                    GlobeImageNum,EffectSrcAlpha);
                  ImageRenderEffectColor(X-GetImageWidth(FlareImageNum) ShR 1,
                                         Y-GetImageHeight(FlareImageNum)   ShR
1,
                                         TransparentLevel ShL 16+
                                         TransparentLevel ShR 8+
                                         TransparentLevel,
                                         FlareImageNum,EffectAdd);
                End;
              Else
                Begin
                  //Unit is selection ?
                  If UnitGroup and 128=128 then
                    Begin
                      Rect(X-_Size2,Y-_Size2,
                           X+_Size2,Y+_Size2,Green,EffectNone);
                      TextOut(X-_Size2,Y-_Size2,
                              Format('HP: %d/%d',[HitPoint,_HitPoint]),White);
                    End;
                  Case UnitCmd of
                    NoCmd,
                    CmdFire,
                    CmdMove,
                    CmdAttack,
                    CmdAttackP,
                    CmdAttackT,
                    CmdAttackM,
                    CmdPatrol,
                    CmdBurn :
                      Begin
                        SprNum:=_ShipAni.Move[CurrentHead].
                                Ani[CurrentFrame].FramePos;
                        Case _ShipAni.Move[CurrentHead].
                             Ani[CurrentFrame].FrameStyle of
                          02 : Effects:=EffectAdd;
                          Else Effects:=EffectSrcAlpha;
                        End;
                        ImageRenderEffect(X-GetImageWidth(_Images[SprNum]) ShR
1,
                                          Y-GetImageHeight(_Images[SprNum])
ShR 1,
                                          _Images[SprNum],Effects);
                      End;
                  End;
                End;
            End;
          If ShowDebug then
            Begin
              If UnitCmd in [CmdMove,CmdAttack,CmdAttackP,
                             CmdAttackT,CmdAttackM,CmdPatrol] then
                Begin
                  TextOut(X,Y,HeadingName[CurrentHead],
                          White,EffectNone);
                  X1:=Round(Pos.X)-GameWorld.ViewPos.X;
                  Y1:=Round(Pos.Y)-GameWorld.ViewPos.Y;
                  X2:=Round(Dest.X)-GameWorld.ViewPos.X;
                  Y2:=Round(Dest.Y)-GameWorld.ViewPos.Y;
                  Line(X1,Y1,X2,Y2,White,EffectNone);
                End;
            End;
          RenderBuffer;
```

```
        End;
  End;

PROCEDURE TGame.RenderInfos;
  Var Tick : LongWord;
  Begin
    Case MouseCommand of
      MouseSelection :
        Begin
          Rect(SelectStart.X,SelectStart.Y,
               SelectEnd.X,SelectEnd.Y,
               White,EffectNone);
        End;
    End;
    If ShowFPS then
      Begin
        Tick:=MMSystem.TimeGetTime-GameStart;
        If Tick<>0 then
          Begin
            TextOut(0,40,Format('UFPS : %0.2f',[GameFrame/Tick*1000]),White);
            TextOut(0,60,Format('SFPS                                    :
%0.2f',[GameScreenFrame/Tick*1000]),White);
            TextOut(0,80,Format('Unit used : %d',[UnitCounting]),White);
          End;
      End;
    //Show mini map ?
    If MiniMapVisible then
      Begin
        Bar(MiniMapPosX-1,MiniMapPosY-1,
            MiniMapPosX+MiniMapSizeX+1,
            MiniMapPosY+MiniMapSizeY+1,
            Gray,EffectAdd);
        ImageRenderEffect(MiniMapPosX,MiniMapPosY,
                          MiniMapImage,EffectAdd);
      End;
    //Show command panel ?
    If CommandPanelVisible then
      Begin
        Bar(PanelPosX-1,PanelPosY-1,
            PanelPosX+PanelSizeX+1,
            PanelPosY+PanelSizeY+1,
            BlueGray,EffectAdd);
        If FirstUnitSelect<>Low(TUnitCount) then
          Begin
            With GameUnits[FirstUnitSelect],
                 UnitsProperty[Typer] do
              Begin
                TextOut(PanelPosX,PanelPosY,
                        ShipName[Typer],White);
                TextOut(PanelPosX,PanelPosY+20,
                        Format('HP: %d/%d',[HitPoint,_HitPoint]),White);
                TextOut(PanelPosX,PanelPosY+40,
                        Format('Damage: %d. Speed: %d',
                               [_Damage,_Speed]),White);
                TextOut(PanelPosX,PanelPosY+60,
                        Format('Range: %d. Gun: %s',
                               [_AttackRange,GunName[_Gun]]),White);
                If _ShipCanCreate<>Low(TSpaceShip) then
                  Begin
                    If CreationCounting=0 then
                      Begin
                        TextOut(PanelPosX+400,PanelPosY,
                                Format('Press C to create %s',
                                       [ShipName[_ShipCanCreate]]),White);
```

```
                    End
                  Else
                    Begin
                      TextOut(PanelPosX+400,PanelPosY,
                              Format('Create %s: %d/%d',
                                     [ShipName[_ShipCanCreate],
                                      CreationCounting,
                                      UnitsProperty[_ShipCanCreate].
                                      _BuildTime]),White);
                    End;
                End;
            End;
        End;
    End;
  //Show player's money and units counting
  TextOut(0,0,Format('Money: %d. Units: %d',
          [GamePlayers[HumanControl].Money,
           GamePlayers[HumanControl].CountUnit]),White);
End;

PROCEDURE TGame.RenderScene;
  Begin
    If GameTime-LastGameScreenTime>ScreenUpdate then
      Begin
        LastGameScreenTime:=GameTime;
        Inc(GameScreenFrame);
        BeginRender;
        RenderWorld;
        RenderUnits;
        RenderInfos;
        EndRender;
      End;
  End;

FUNCTION  TGame.GetShipTyper(Name : String) : TSpaceShip;
  Var Z : TSpaceShip;
  Begin
    For Z:=Low(TSpaceShip) to
           High(TSpaceShip) do
      If ShipName[Z]=Name then
        Begin
          Result:=Z;
          Exit;
        End;
    Result:=Low(TSpaceShip);
  End;

FUNCTION  TGame.GetPlayer(Name : String) : TPlayer;
  Var Z : TPlayer;
  Begin
    For Z:=Low(TPlayer) to
           High(TPlayer) do
      If PlayerName[Z]=Name then
        Begin
          Result:=Z;
          Exit;
        End;
    Result:=Low(TPlayer);
  End;

FUNCTION  TGame.GetHeading(Name : String) : THeading;
  Var Z : THeading;
  Begin
    For Z:=Low(THeading) to
```

```
            High(THeading) do
      If HeadingName[Z]=Name then
        Begin
          Result:=Z;
          Exit;
        End;
    Result:=Low(THeading);
  End;

FUNCTION  TGame.GetGun(Name : String) : TGun;
  Var Z : TGun;
  Begin
    For Z:=Low(TGun) to
           High(TGun) do
      If GunName[Z]=Name then
        Begin
          Result:=Z;
          Exit;
        End;
    Result:=Low(TGun);
  End;
END.
```

Chapter IV: Sumary

Final ?

Như vậy là chúng ta đã có một game RTS (xạo) không hoàn chỉnh. Với game này mà tung ra "thị trường" thì cầm chắc là…chít thảm, he he. Nhưng hi vọng với code này các bạn có thể hiểu sơ sơ việc viết mã cho một game thì phân rã như thế nào. Với game này chưa thể gọi là một game RTS, hơn nữa do thời gian phát triển quá ít nên nó sẽ chứa đầy lỗi. Như vậy công việc của bạn bây giờ là sửa lỗi và nâng cấp trên khung này.

Với chương trình này, hiện tại tôi vẫn chưa comment một cách đầy đủ vì thiếu thời gian, nhưng tôi nghĩ rằng các bạn có thể hiểu được vì cách viết của tôi khá dễ hiểu. Nếu không hiểu rõ lắm, hãy mail cho tôi, tôi sẽ dành thời gian chú thích rõ ràng hơn.

Upgrade

Mặc dù không test nhiều tui cũng chắc chắn chương trình chứa một số lỗi như sau:

- Có lúc kéo chuột để chọn quân lại không được (Kì chưa ?).
- Có lúc đặt lệnh (không nhớ lệnh gì) thì các unit lại chạy ra một chỗ khác (Kì nữa ?)
- ….Chắc là còn nhìu lỗi nữa tui chưa tìm ra….

Công việc của bạn là sửa những lỗi này và tối ưu hoá chương trình. Cạnh đó tôi nghĩ chương trình còn cần nâng cấp một số tính năng như sau:

- Mặc dù khi đặt lệnh cho một nhóm quân (Move, Patrol…) tôi đã cố tình đặt cho vị trí đích của lệnh lệch đi một chút, nhưng chắc chắn sẽ có trường hợp các unit sẽ nằm đè rít rịt lên nhau (hic). Bạn cần thêm tính năng tự "dãn" ra cho các đơn vị này. Gợi ý: Hix, kiểm soát số lượng các unit bị coi là "chồng nhau" chăng ? Có thể được nhưng cách này thực thi khá chậm.
- Toàn bộ các đơn vị trên bản đồ đều bị nhìn thấy (?) tức là chương trình chưa tính đến "fog of war". Gợi ý: cập nhật theo thời gian như MiniMap, những unit nào trong

tầm nhìn của đơn vị thuộc phe người chơi thì đánh dấu nó lại. Vấn đề là phải viết hàm này sao cho nhanh (hix).

- Các đơn vị trong game giới hạn chỉ có thể sinh một loại quân hoặc không, tính năng này có thể nâng cấp thành cho phép chọn loại quân có thể sinh. Bạn cần thêm một list các loại quân có thể sinh vào dữ liệu đặc tính của unit trong game, sau đó xây dựng hệ thống menu cho game để người chơi có thể lựa chọn những tính năng sinh quân này. Các tính năng như di chuyển, tấn công cũng cần được hiển thị rõ ràng hơn thông qua hệ thống menu.
- Trong game chưa có khai thác tài nguyên, thêm công việc này sẽ khiến game có tính cạnh tranh hơn, mặc dù như vậy sẽ khó tính toán AI cho máy hơn và nhiều game hiện tại thường giảm nhẹ sự quan tâm đến vấn đề này (?).
- Nâng cấp tính năng AI: Hiện tại game hoàn toàn chưa có AI, các đơn vị quân máy chỉ đơn giản là sinh quân ầm ầm rùi chạy toán loạn quanh bản đồ. Bạn nên xây dựng modun cho phép máy điều khiển quân tập trung lại rùi "lùng giết" quân của người chơi sẽ hiệu quả hơn.
- Hoàn thiện tính năng âm thanh của game. Game cần xây dựng một modun quản lý âm thanh riêng, các thực thi âm thanh được gọi nên có các thông số đại loại như vị trí, kiểu âm thanh… nhằm xác định độ to nhỏ và loại bỏ bớt các âm thanh cùng loại được gọi liên tục, ví dụ như một đám tàu cùng bắn tên lửa chẳng hạn, modun này có thể chỉ phát 2/3 âm thanh bắn mà cũng đạt được hiệu quả tương đương.
- ….Tiếp tục là tùy ý bạn….Hix, công việc vẫn còn rất nhìu ở phía trước hix hix…

Dưới đây là screen shot từ một game RTS tôi đang phát triển:

Reference

AvenusHelper

- Định nghĩa kiểu:
  - TBitDepth: Kiểu độ sâu màu của màn hình.
    - BD16Bit: Độ sâu màu 16bit.
    - BD32Bit: Độ sâu màu 32bit.
  - TImageFormat: Kiểu format của ảnh.
    - FormatR3G3B2: Ảnh 8bit (Red 3bit, Green 3bit, Blue 2bit).
    - FormatX4R4G4B4: Ảnh 16bit (Mask 4bit, Red 4bit, Green 4bit, Blue 4bit).
    - FormatA4R4G4B4: Ảnh 16bit (Alpha 4bit, Red 4bit, Green 4bit, Blue 4bit).
    - FormatA8R3G3B2: Ảnh 16bit (Alpha 8bit, Red 3bit, Green 3bit, Blue 2bit).
    - FormatR5G6B5: Ảnh 16bit (Red 5bit, Green 6bit, Blue 5bit).
    - FormatX1R5G5B5: Ảnh 16bit (Mask 1bit, Red 5bit, Green 5bit, Blue 5bit).
    - FormatA1R5G5B5:Ảnh 16bit (Alpha 1bit, Red 5bit, Green 5bit, Blue 5bit).
    - FormatX8R8G8B8: Ảnh 32bit (Mask 8bit, Red 8bit, Green 8bit, Blue 8bit).
    - FormatA8R8G8B8:Ảnh 32bit (Alpha 8bit, Red 8bit, Green 8bit, Blue 8bit).

- Định nghĩa hằng:
  - EffectNone: Vẽ không hiệu ứng.
  - EffectAdd: Vẽ kiểu cộng.
  - EffectSrcAlpha: Vẽ theo kênh alpha của ảnh, chỉ áp dụng với những ảnh có kênh alpha.
  - EffectSrcAlphaAdd: Vẽ kiểu cộng theo kênh alpha của ảnh, chỉ áp dụng với những ảnh có kênh alpha.
  - EffectMul: Vẽ theo kiểu nhân.
  - EffectSrcColor: Vẽ sử dụng màu source như là kênh alpha.
  - EffectSrcColorAdd: Vẽ kiểu cộng, sử dụng màu source như là kênh alpha.
  - EffectInvSrcColor: Vẽ với giá trị màu nguồn đảo ngược.
  - EffectInvSrcAlpha: Vẽ với giá trị kênh alpha đảo ngược.

- EffectInvSrcMask: Vẽ với giá trị mask đảo ngược.
- EffectBlend1, EffectBlend2, EffectBlend3, EffectBlend4: Các kiểu hiệu ứng blend ảnh đang thử nghiệm.
- EffectShadow: Vẽ bóng.
- EffectBlendColor: Vẽ blend với màu nguồn.
- EffectBlendColorAlphaChannel: Vẽ blend sử dụng kênh alpha.
- EffectAddDiffuse, EffectDiffuseAlpha: Vẽ khuếch tán với kênh alpha.
- EffectMirror, EffectFlip: Vẽ lật hình, hiệu ứng này có thể kết hợp với các hiệu ứng ở trên.
- Các hằng số Key_# là các hằng số định nghĩa số hiệu phím, bạn có thể dễ dàng đoán ra nó đại diện cho phím nào, tôi không chú thích ở đây do nó dài quá .

- Các hàm, lệnh:
  - Hàm CreateWindow(Width,Height : Integer;Title : PChar) : Boolean
    - Tạo một cửa sổ kích thước Width x Height có caption là Title.
    - Hàm trả về true nếu tạo thành công, trả về false nếu không tạo được cửa sổ.
  - Hàm CreateFullScreen(Width,Height : Integer;ScreenDepth : TBitDepth;Title : PChar) : Boolean
    - Khởi tạo chế độ màn hình Width x Height độ sâu màu là ScreenDepth có caption là Title.
    - Hàm trả về true nếu khởi tạo thành công, trả về false nếu không khởi tạo được chế độ đó.
  - Lệnh DestroyScreen
    - Huỷ cửa sổ/chế độ đã tạo trước đó, sau khi gọi lệnh này tất cả các hàm vẽ, lấy phím… đều sinh lỗi do chương trình đã huỷ handle quản lý tất cả các dịch vụ này.
  - Hàm EventMessage : Boolean
    - Kiểm tra các events, trả về false nếu xuất hiện event Quit.
  - Lệnh LoadFont(FileName : String;SizeX,SizeY,Width,Height,TransColor : LongWord)
    - Đọc font chữ từ file ảnh FileName
  - Lệnh TextOut(X,Y : Integer;Text : String;Color : LongWord)
    - Viết xâu Text ra màn hình, ở toạ độ [X,Y], màu Color.
  - Lệnh TextOut(X,Y : Integer;Text : String;Color : LongWord;Effect : Integer)
    - Viết xâu Text ra màn hình, ở toạ độ [X,Y], màu Color. Hiệu ứng Effect
  - Hàm CreateImage(SizeX,SizeY : Integer;Format : TImageFormat) : Integer
    - Tạo một ảnh có kích thước SizeX x SizeY với format là Format.
    - Số trả về là số hiệu của ảnh đã tạo.
  - Hàm LoadImage(FileName : String;Format : TImageFormat) : Integer
    - Đọc ảnh từ FileName, ảnh đọc vào với format là Format.
    - Hỗ trợ các định dạng ảnh JPEG, PNG, BMP.
    - Số trả về là số hiệu của ảnh đã đọc.
  - Hàm SetAlphaMask(ImageNum,TransparentColor : Integer) : Integer
    - Đặt kênh Alpha cho tất cả các điểm ảnh có màu Transparent.
    - Chỉ hỗ trợ format ảnh FormatA1R5G5B5.
  - Hàm GetImageWidth(ImageNum : Integer) : Integer
    - Lấy độ rộng của ảnh số hiệu ImageNum.
  - Hàm GetImageHeight(ImageNum : Integer) : Integer
    - Lấy độ cao của ảnh số hiệu ImageNum.
  - Hàm ImageLock(ImageNum : Integer) : Integer

        - Lock ảnh số hiệu ImageNum.
    - Hàm ImageUnLock(ImageNum : Integer) : Integer
        - UnLock ảnh số hiệu ImageNum.
    - Lệnh ImageClear(ImageNum : Integer;Color : Cardinal)
        - Xoá ảnh ImageNum bằng màu Color.
    - Lệnh ImagePutPixel(ImageNum,X,Y : Integer;Color : Cardinal)
        - Vẽ điểm ảnh màu Color ở toạ độ [X,Y] trong ảnh ImageNum.
    - Hàm ImageGetPixel(ImageNum,X,Y : Integer) : Cardinal
        - Lấy màu điểm ảnh Color ở toạ độ [X,Y] trong ảnh ImageNum.
    - Lệnh ImageLine(ImageNum,X1,Y1,X2,Y2 : Integer;Color : Cardinal)
        - Vẽ đường thẳng màu Color từ [X1,Y1] đến [X2,Y2] trong ảnh ImageNum.
    - Lệnh ImageRect(ImageNum,X1,Y1,X2,Y2 : Integer;Color : Cardinal)
        - Vẽ hình chữ nhật màu Color từ [X1,Y1] đến [X2,Y2] trong ảnh ImageNum.
    - Lệnh ImageFillRect(ImageNum,X1,Y1,X2,Y2 : Integer;Color : Cardinal)
        - Tô hình chữ nhật màu Color từ [X1,Y1] đến [X2,Y2] trong ảnh ImageNum.
    - Lệnh BeginRender
        - Bắt đầu vẽ scene mới (xoá buffer, xoá render buffer).
    - Lệnh EndRender
        - Kết thúc vẽ scene.
    - Lệnh ClearScreen(Color : Cardinal)
        - Xoá màn hình bằng màu Color.
    - Hàm RenderBuffer : Integer
        - Dựng toàn bộ hình trong render buffer.
    - Lệnh SetClipRect(X1,Y1,X2,Y2 : Integer)
        - Đặt clip rect.
    - Lệnh Rect(X1,Y1,X2,Y2 : Integer;Color : Cardinal;Effect : Integer)
        - Vẽ hình chữ nhật.
    - Lệnh Bar(X1,Y1,X2,Y2 : Integer;Color : Cardinal;Effect : Integer)
        - Tô hình chữ nhật.
    - Lệnh Line(X1,Y1,X2,Y2 : Integer;Color : Cardinal;Effect : Integer)
        - Vẽ đường thẳng.
    - Lệnh SmoothLine(X1,Y1,X2,Y2 : Integer;Color : Cardinal)
        - Vẽ đường thẳng smooth (mịn).
    - Hàm ImageRenderEffect(XPos,YPos,Pattern,Effect : Integer) : Boolean
        - Dựng ảnh Pattern ở vị trí [XPos, YPos] với hiệu ứng Effect.
    - Hàm ImageRenderEffect(X1,Y1,X2,Y2,Pattern,Effect : Integer) : Boolean
        - Dựng ảnh Pattern thành hình chữ nhật [X1,Y1,X2,Y2] với hiệu ứng Effect.
    - Hàm ImageRenderEffect(XPos,YPos,Scale,Pattern,Effect : Integer) : Boolean
        - Dựng ảnh Pattern ở vị trí [XPos, YPos] với hiệu ứng Effect. Tỉ lệ zoom là Scale/256.
    - Hàm ImageRenderEffectColor(XPos,YPos : Integer;Color : Cardinal;Pattern,Effect : Integer) : Boolean
        - Dựng ảnh Pattern ở vị trí [XPos, YPos] với hiệu ứng Effect. Màu áp là Color.
    - Hàm ImageRenderEffectColor(X1,Y1,X2,Y2 : Integer;Color : Cardinal;Pattern,Effect : Integer) : Boolean
        - Dựng ảnh Pattern thành hình chữ nhật [X1,Y1,X2,Y2] với hiệu ứng Effect. Màu áp là Color.

- Hàm ImageRenderEffectColor(XPos,YPos : Integer;C1,C2,C3,C4 : LongWord;Pattern,Effect : Integer) : Boolean
  - Dựng ảnh Pattern ở vị trí [XPos, YPos] với hiệu ứng Effect. Màu áp ở bốn góc lần lượt là C1, C2, C3, C4.
- Hàm ImageRenderEffectColor(XPos,YPos,Scale : Integer;Color : Cardinal;Pattern,Effect : Integer) : Boolean
  - Dựng ảnh Pattern ở vị trí [XPos, YPos] với hiệu ứng Effect. Tỉ lệ zoom là Scale/256. Màu áp là Color.
- Lệnh GetInputStatus
  - Cập nhật input (Trạng thái mouse, keyboard…)
- Hàm KeyDown(Key : Byte) : Boolean
  - Kiểm tra phím Key có bị nhấn hay không ?
- Hàm KeyPress(Key : Byte) : Boolean
  - Kiểm tra phím Key có bị gõ hay không ?
- Hàm MouseFirstClickL : Boolean
  - Click chuột trái ?
- Hàm MouseFirstClickR : Boolean
  - Click chuột phải ?
- Hàm MouseFirstClickM : Boolean
  - Click chuột giữa ?
- Hàm MouseReleasedL : Boolean
  - Thả chuột trái ?
- Hàm MouseReleasedR : Boolean
  - Thả chuột phải ?
- Hàm MouseReleasedM : Boolean
  - Thả chuột giữa ?
- Hàm MouseHoldL : Boolean
  - Nhấn chuột trái ?
- Hàm MouseHoldR : Boolean
  - Nhấn chuột phải ?
- Hàm MouseHoldM : Boolean
  - Nhấn chuột giữa ?
- Hàm MouseX : Integer
  - Toạ độ X của chuột.
- Hàm MouseY : Integer
  - Toạ độ Y của chuột.
- Hàm CreateSound(FileName : String;Loop : Boolean) : Integer
  - Tạo handle cho sound FileName.
  - Trả về handle quản lý sound này.
- Lệnh SoundPlay(Slot : Integer)
  - Chơi sound số hiệu slot.
- Lệnh SoundPause(Slot : Integer)
  - Tạm dừng sound số hiệu slot.
- Lệnh SoundStop(Slot : Integer)
  - Dừng sound số hiệu slot.
- Lệnh SoundSetVolume(Slot : Integer;Value : Byte)
  - Đặt cường độ âm thanh cho Slot.
- Hàm SoundGetVolume(Slot : Integer) : Byte
  - Lấy cường độ âm thanh của Slot.