```
; Sourcecode snippets from Transport Tycoon Deluxe (DOS PC)
; (C) 1994-1995 Chris Sawyer, All Rights Reserved


HillSpeed PROC
; See if loco, and slow it down/speed it up if on hill
;  dl=U coord before moving
;  esi->Obj data
; Alters: edx
      cmp dl,[esi].Obj.U            ; see if height changed
      je hs2_level                  ;
      mov dx,[esi].Obj.Speed        ; uphill
      ja hs2_downhill               ;
      shr dx,2                      ;
      sub [esi].Obj.Speed,dx        ;
      ret                           ;
hs2_downhill:
      add dx,2                      ; faster
      cmp dx,[esi].Obj.MaxSpeed     ;
      ja hs2_level                  ;
      mov [esi].Obj.Speed,dx        ;
hs2_level:
      ret                           ;
HillSpeed ENDP


IncreaseSpeed PROC
; Increase loco speed to maximum, calculate whether should move on
screen
;  esi->Obj data
; Returns:
;  ah=no. of unit moves to do
; Alters: eax,ebx,es
      movzx ax,[esi].Obj.Acceleration ; 64=accel. of 1 etc
      shl ax,2                      ;
      mov bl,al                     ; fraction
      shr ax,8                      ; whole
      add [esi].Obj.SpeedFraction,bl      ;
      adc ax,[esi].Obj.Speed        ;
      cmp ax,[esi].Obj.MaxSpeed     ;
      jbe @F                        ;
      mov ax,[esi].Obj.MaxSpeed     ;
@@:   mov [esi].Obj.Speed,ax        ;
      test [esi].Obj.Dir,bit0       ; see if moving diagonally
      jnz @F                        ;
      mov bx,ax                     ; 3/4 speed for diagonals
      shl bx,1                      ;
      add ax,bx                     ;
      shr ax,2                      ;
@@:   sub [esi].Obj.Delay,al        ;
      adc ah,0                      ;
      ret                           ;
IncreaseSpeed ENDP


SmokeEffects PROC
;  esi->Obj data
; Preserves: si
      DebugStamp 14020253h          ;
      bt [esi].Obj.Flags,ObjFlags.ForbidSmoke ;
      jc se_exit                    ; smoke forbidden (train slowing)
      mov al,[esi].Obj.VehType      ;
      cmp al,TrainVehType.SteamLocos      ;
      jb se_steamsmoke              ;
```

```
      cmp al,TrainVehType.DieselLocos ;
      jb se_dieselsmoke               ;
      cmp al,TrainVehType.ElectricLocos ;
      jb se_electricspark             ;
se_exit:
      ret                             ;

se_electricspark:
      test [esi].Obj.GeneralCount,7 ;
      jnz se_exit                   ;
      cmp [esi].Obj.Speed,2         ;
      jb se_exit                    ;
      cmp [esi].Obj.Halted,0        ;
      jne se_exit                   ;
      test [esi].Obj.Flags,1 shl ObjFlags.Invisible ;
      jnz se_exit                   ; - invisible
      test [esi].Obj.TrainTrackType,TTTunnel+TTBuilding ;
      jnz se_exit                   ; - building or tunnel
      mov bx,[esi].Obj.BlockAddress ; check for being in loco depot
      mov al,fs:[bx]                ;
      and al,11110000b              ;
      cmp al,Mno.Rail*2             ;
      jne @F                        ;
      mov al,gs:[bx]                ;
      and al,11111100b              ;
      cmp al,11000000b              ;
      je se_exit                    ; - in depot
@@:   call Random                   ;
      cmp ax,10000h/45              ; random spark occurance
      ja se_exit                    ;
      push esi                      ;
       mov ax,[esi].Obj.L           ;
       mov cx,[esi].Obj.R           ;
       mov dl,[esi].Obj.U           ;
       add dl,10                    ; height above locomotive
       mov di,EffectType.TrainSpark ;
       ModuleCall Effect,General,CreateEffectObj ; create spark effect
      pop esi                       ;
      ret                           ;

se_steamsmoke:
      test [esi].Obj.GeneralCount,31      ;
      jnz se_nosmoke                ;
      cmp [esi].Obj.Speed,2         ;
      jb se_nosmoke                 ;
      cmp [esi].Obj.Halted,0        ;
      jne se_nosmoke                ; no smoke if stopped
      test [esi].Obj.Flags,1 shl ObjFlags.Invisible ;
      jnz se_nosmoke                ; no smoke if invisible
      test [esi].Obj.TrainTrackType,TTTunnel+TTBuilding ;
      jnz se_nosmoke                ; no smoke if in building or tunnel
      mov bx,[esi].Obj.BlockAddress ; check for being in loco depot
      mov al,fs:[bx]                ;
      and al,11110000b              ;
      cmp al,Mno.Rail shl 1         ;
      jne @F                        ;
      mov al,gs:[bx]                ;
      and al,11111100b              ;
      cmp al,11000000b              ;
      je se_nosmoke                 ; don't steam in depot
@@:   mov al,fs:[bx]                ; check for being in tunnel mouth
      and al,11110000b              ;
      cmp al,Mno.CivEng shl 1       ;
      jne @F                        ;
```

```
      test BYTE PTR gs:[bx],bit7     ;
      jz se_nosmoke                  ;
@@:   push esi                       ;
       mov ax,[esi].Obj.L            ;
       mov cx,[esi].Obj.R            ;
       movzx ebx,[esi].Obj.Dir       ;
       add ax,SmokePos[ebx*4]        ;
       add cx,SmokePos[ebx*4+2]      ;
       mov dl,[esi].Obj.U            ;
       add dl,10                     ;
       mov di,EffectType.TrainSteam  ;
       ModuleCall Effect,General,CreateEffectObj ; create steam effect
      pop esi                        ;
se_nosmoke:
      ret                            ;

se_dieselsmoke:
      cmp [esi].Obj.Speed,2          ; no smoke if stationary
      jb se_nodsmoke                 ;
      cmp [esi].Obj.Speed,40         ; no smoke if going fast
      ja se_nodsmoke                 ;
      cmp [esi].Obj.Halted,0         ;
      jne se_nodsmoke                ; no smoke if stopped
      test [esi].Obj.Flags,1 shl ObjFlags.Invisible ;
      jnz se_nodsmoke2               ; no smoke if invisible
      test [esi].Obj.TrainTrackType,TTTunnel+TTBuilding ;
      jnz se_nodsmoke2               ; no smoke if in building or tunnel
      mov bx,[esi].Obj.BlockAddress  ; check for being in loco depot
      mov al,fs:[bx]                 ;
      and al,11110000b               ;
      cmp al,Mno.Rail shl 1          ;
      jne @F                         ;
      mov al,gs:[bx]                 ;
      and al,11111100b               ;
      cmp al,11000000b               ;
      je se_nodsmoke2                ; don't smoke in depot
@@:   call Random                    ;
      cmp ax,1e00h                   ;
      ja se_nodsmoke2                ;
      push esi                       ;
       mov ax,[esi].Obj.L            ;
       mov cx,[esi].Obj.R            ;
       mov dl,[esi].Obj.U            ;
       add dl,10                     ;
       mov di,EffectType.TrainSmoke  ;
       ModuleCall Effect,General,CreateEffectObj ; create smoke effect
      pop esi                        ;
se_nodsmoke2:
      movzx ebx,[esi].Obj.DesignType       ; check for loco pair
      test TrainSpecials[ebx-ObjDesign.Trains],bit0 ;
      jz se_nodsmoke                 ;
      push esi                       ;
se_dsmokelp:
       mov ax,[esi].Obj.Trailer      ; find 2nd loco
       cmp ax,-1                     ;
       je se_dsmokefnd               ;
       GetObjAddr esi,ax             ;
       jmp se_dsmokelp               ;
se_dsmokefnd:
       test [esi].Obj.Flags,1 shl ObjFlags.Invisible ;
       jnz se_nodsmoke3              ; no smoke if invisible
       test [esi].Obj.TrainTrackType,TTTunnel+TTBuilding ;
       jnz se_nodsmoke3              ; no smoke if in building or tunnel
```

```
        mov bx,[esi].Obj.BlockAddress      ; check for being in loco
depot
        mov al,fs:[bx]                  ;
        and al,11110000b                ;
        cmp al,Mno.Rail*2               ;
        jne @F                          ;
        mov al,gs:[bx]                  ;
        and al,11111100b                ;
        cmp al,11000000b                ;
        je se_nodsmoke3                 ; don't smoke in depot
@@:     call Random                     ;
        cmp ax,1e00h                    ;
        ja se_nodsmoke3                 ;
        mov ax,[esi].Obj.L              ;
        mov cx,[esi].Obj.R              ;
        mov dl,[esi].Obj.U              ;
        add dl,10                       ;
        mov di,EffectType.TrainSmoke    ;
        ModuleCall Effect,General,CreateEffectObj ; create smoke effect
se_nodsmoke3:
      pop esi                           ;
se_nodsmoke:
      ret                               ;
SmokeEffects ENDP


CalcDirectionToNewBlock PROC
; Calculate direction from old block to new block, based upon the
; old and new block addresses
;  bx=old block address
;  di=new block address
; Returns:
;  ebp=Direction
; Alters: ebx,dx,bp
      mov dx,di                         ;
      sub bh,dh                         ;
      sub bl,dl                         ;
      inc bl                            ;
      inc bh                            ;
      shl bh,2                          ;
      or bl,bh                          ;
      and ebx,0ffh                      ;
      movzx ebp,CalcDirection[ebx]      ;
      xor bp,4                          ; ebp=Direction
      ret                               ;
CalcDirectionToNewBlock ENDP


GetPossibleRoutes PROC
; Get possible rail routes through new block
;  edi=new block address
;  ebp=Direction into new block
; Returns:
;  dl=TrackType bits (set for each possible TrackType)
;  dh=TrackType bits (set for each blocked TrackType)
;  z flag set for state of dl
; Alters: edx
      mpush eax,esi,ebp                 ;
        mov ax,RouteType.Rail           ; look for rail routes
        call GetRoutes                  ; get routes across new block
       pop ebp                          ;
       mov dx,ax                        ;
       shr eax,16                       ;
```

```
        and dx,LegalRoutes2[ebp-1]    ; get possible routes for this
direction
        and ax,LegalRoutes2[ebp-1]    ;
        or dl,dh                      ; available routes
        mov dh,al                     ;
        or dh,ah                      ; blocked routes
       mpop esi,eax                   ;
       or dl,dl                       ;
       ret                            ;
GetPossibleRoutes ENDP


SetRouteAndPosition PROC
; Choose a route across new block, and calculate L,R coords for new
position
; Also checks for blockages on chosen route (signal red etc.)
;  esi->Obj data
;  ax,cx=new L,R coordinates
;  di=new block address
;  dl=TrackType bits (set for each possible TrackType)
;  dh=TrackType bits (set for each blocked TrackType)
;  ObjectTower->tower Obj (0=no tower)
;  ebp=Direction of edge train has just crossed
; Returns:
;  ax,cx=new L,R coordinates
;  dl=Direction onto new block
;  di=TrackType chosen in new block
;  bp=new block address
;  c flag set if chosen route blocked
; Alters: eax,ebx,ecx,edx,edi,ebp,es
       push di                        ; save new block addess
        mov ebx,ObjectTower           ; check for following tower
        or ebx,ebx                    ;
        jnz srap_towed                ; - must follow tower

        push dx                       ;
         call ChooseRoute             ; get chosen TrackType in di
        pop bx                        ; bh=blocked TrackTypes
srap_gotit:
        push di                       ; di=TrackType
         mov dx,di                    ;
         and edi,0ffffh               ;
         shl di,1                     ; di=3*route no.
         add di,dx                    ;
         and al,NOT(BlSize-1)         ;
         and cl,NOT(BlSize-1)         ;
         mov ebp,EdgeStarts[(ebp*2)-2] ;
         add al,[ebp+edi]             ;
         add cl,[ebp+edi+1]           ;
         movzx dx,BYTE PTR [ebp+edi+2] ; direction
       mpop di,bp                     ; new block address
       mov bl,bh                      ; blocked TrackTypes
       bt bx,di                       ; see if TrackType blocked
       ret                            ;

srap_towed:
        mpush ax,cx                   ;
         sub ax,[ebx].Obj.L           ; calc vector from tower to trailer
         js srap_1                    ;
         cmp al,2                     ;
         mov al,-1                    ;
         jg srap_2                    ;
         xor al,al                    ;
         jmp srap_2                   ;
```

```
srap_1:
        cmp al,-2                       ;
        mov al,1                        ;
        jl srap_2                       ;
        xor al,al                       ;
srap_2:
        sub cx,[ebx].Obj.R              ;
        js srap_1a                      ;
        cmp cl,2                        ;
        mov cl,-1                       ;
        jg srap_2a                      ;
        xor cl,cl                       ;
        jmp srap_2a                     ;
srap_1a:
        cmp cl,-2                       ;
        mov cl,1                        ;
        jl srap_2a                      ;
        xor cl,cl                       ;
srap_2a:
        inc al                          ;
        inc cl                          ;
        mov bl,al                       ;
        shl cl,2                        ;
        or bl,cl                        ;
        and ebx,0ffh                    ;
        mov bl,CalcDirection[ebx]       ; bl=Direction to follow tower
        and bl,3                        ;
        mov bl,TrackTypesForDirection[ebx] ; bl=valid track types
       mpop cx,ax                       ;
       xor bh,bh                        ; not blocked
       and bl,dl                        ; see if can go this way
       jnz srap_notstuck                ;
       not bh                           ; blocked
       mov bl,dl                        ; use a valid direction
srap_notstuck:
       bsf di,bx                        ; get TrackType in di
       jmp srap_gotit                   ;
SetRouteAndPosition ENDP


CalcVisualDirection PROC
; Calulate the visual direction of an object from the current and
previous
; L,R coordinates
;  esi->Obj (old L,R position in Obj.L,Obj.R)
;  ax,cx=new L,R position
; Returns:
;  bx=Direction
; Alters: ebx,dx
      movzx ebx,ax                      ;
      mov dx,cx                         ;
      sub bx,[esi].Obj.L                ; get L,R incs
      sub dx,[esi].Obj.R                ;
      inc bx                            ; 0..2 (bh=0)
      inc dl                            ; 0..2
      shl dl,2                          ;
      or bl,dl                          ;
      mov bl,CalcDirection[ebx]         ; get Direction (bh=0)
      ret                               ;
CalcVisualDirection ENDP


SetNewPosition PROC
; Set new L,R,U coords, and update object on screen
```

```
;  esi->Obj data
;  ax,cx=new L,R coords
; Returns:
;  dl=old U coord
; Alters: eax,ecx,edx,edi,ebp,es
      push bx                         ; save direction for trailer
       mov [esi].Obj.L,ax             ;
       mov [esi].Obj.R,cx             ;
       mov dl,[esi].Obj.U             ; old U coord
       push dx                        ;
        push esi                      ;
         call GetHeight               ;
        pop esi                       ;
        test dh,bit0                  ; check for bi-level block
        jz snp_ok                     ;
        cmp dl,[esi].Obj.U            ;
        je snp_ok                     ;
        add dl,BlHeight               ; upper level
snp_ok:
        mov [esi].Obj.U,dl            ;
        call CalcObjectArea           ;
        call UpdateOldAndNewArea      ;
       pop dx                         ;
      pop bx                          ; get increments
      ret                             ;
SetNewPosition ENDP


CheckNewStation PROC
; Check to see if this is first train to stop at this station
; If so, display a message
;  esi->Obj data
;  al=station no.
; Preserves: esi
      and eax,0ffh                    ;
      imul ax,StationData             ;
      add eax,Stations                ; eax->StationData
      bts [eax].StationData.Flags,StationFlags.FirstTrain ;
      jnc cns_firsttrain              ;
      ret                             ;
cns_firsttrain:
      mov ebx,MessClass.FirstArrivalPlayer shl 16 ;
      mov dl,[esi].Obj.Owner          ;
      cmp dl,Player1                  ;
      je @F                           ;
      mov ebx,MessClass.FirstArrivalNonPlayer shl 16 ;
@@:
      mov dx,[eax].StationData.NameStr ;
      mov TextParams[0],dx            ;
      mov edi,[eax].StationData.Town      ;
      mov edx,[edi].TownData.Seed     ;
      mov DWORD PTR TextParams[4],edx ;
      mov dx,[edi].TownData.NameStr ;
      mov TextParams[2],dx            ;
      mov ax,[esi].Obj.Num            ;
      mov MessView1,ax                ;
      mov bx,MessType.NewspaperSmall+(((1 shl MessFlags.View3DObject)+(1
shl MessFlags.ClickObj)) shl 8) ;
      mov dx,Sno.Train.FirstTrainStation ;
      jmp ShowMessage                 ;
CheckNewStation ENDP


DoLoco PROC
```

```
; Called twice per game cycle for all trains
; Move loco (and trailers)
;  esi->Obj (loco)
; Alters: eax,ebx,ecx,edx,edi,esi,ebp
      DebugStamp 14020254h          ;
      inc [esi].Obj.GeneralCount    ; general use counter

      cmp [esi].Obj.TrainCrash,0    ; see if crashing
      jne DoLocoCrash               ;

      cmp [esi].Obj.TrainPassSignal,0 ; see if 'pass red signal' flag
set
      je @F                         ;
      dec [esi].Obj.TrainPassSignal ;
@@:
      cmp [esi].Obj.BreakdownState,0      ; see if breaking down
      je dl_notbreakdown            ;
      cmp [esi].Obj.BreakdownState,2      ;
      jbe LocoBreakdown             ;
      dec [esi].Obj.BreakdownState  ;
dl_notbreakdown:

      test [esi].Obj.Flags,1 shl ObjFlags.Stopped ; see if train stopped
      jnz dl_donetrain              ;

      call SmokeEffects             ; smoke/steam animation etc.
      call CheckCurrentOrder        ; ensure current order valid
      cmp TrainTurn,0               ;
      jne dl_reverse                ;
      call ExecuteCurrentOrder      ; execute current order

      mov ax,[esi].Obj.Action       ; get current order
      and al,01fh                   ;
      cmp al,Order.LeavingStation   ;
      je @F                         ;
      cmp al,Order.GoToDest         ;
      jae dl_donetrain              ; not moving
@@:
      call CheckExitDepot           ; deal with trains exiting depot
      call IncreaseSpeed            ; increase loco speed to maximum
      or ah,ah                      ;
      jz dl_donetrain               ; - not moved on screen
      push ax                       ;
       call CheckApplyBrakes        ; slow down for signals etc.
      pop ax                        ;
dl_moveloop:
      mpush ax,esi                  ;
       call MoveTrain               ;
      mpop esi,ax                   ;
      cmp [esi].Obj.Speed,100h      ;
      jbe @F                        ;
      dec ah                        ;
      jnz dl_moveloop               ;
@@:   ret                           ;
dl_reverse:
      mov [esi].Obj.Halted,0        ; reset waiting counter
      mov [esi].Obj.Speed,0         ;
      mov [esi].Obj.SpeedFraction,0 ;
      call ReverseTrain             ; reverse entire train
dl_donetrain:
      ret                           ;
DoLoco ENDP
```

```
MoveTrain PROC
;  esi->Obj (loco)
      mov ObjectTower,0              ; flag this is loco
mt_trailer:
      call NoteScreenArea            ; save old update position
      cmp [esi].Obj.TrainTrackType,1 shl TrackType.Tunnel ;
      je mt_intunnel                 ;
      call CalcNewPos                ; calc new coords, block address
      jne mt_newblock                ;
      cmp [esi].Obj.TrainTrackType,1 shl TrackType.Building ; see if in
depot
      jne mt_notindepot              ;
      mov ax,[esi].Obj.L             ; use current position
      mov cx,[esi].Obj.R             ;
      jmp mt_updatepos               ; don't do interaction check
mt_notindepot:
      mov bp,di                      ; block address
      call ObjectBlockInteraction    ; interact with block at proposed
new position
      or ebp,ebp                     ; see if move forbidden
      js mt_endoftrack               ;
      test ebp,bit29                 ; see if train at station
      jz mt_notstation               ;
      mpush ax,bx                    ;
       mov al,BYTE PTR Temp[0]       ;
       mov [esi].Obj.LastStation,al  ; note last station stopped at
       call CheckNewStation          ; display message for new station
      mpop bx,ax                     ;
      mov ax,[esi].Obj.Action        ;
      mov [esi].Obj.Action,Order.LoadingAtStation ;
      mov dl,al                      ;
      and dl,01fh                    ;
      cmp dl,Order.GoToStation       ;
      jne mt_donestation             ;
      cmp ah,BYTE PTR Temp[0]        ; see if destination station
      jne mt_donestation             ;
      or [esi].Obj.Action,bit7       ; set 'reached dest' bit
      and ax,bit5+bit6               ; get 'full load/empty load' bit
      or [esi].Obj.Action,ax         ;
mt_donestation:
      mov ExpendType,Expenditure.TrainIncome ;
      call LoadVehicles              ; get cargo, set load time
      or al,al                       ; see if anything loaded/unloaded
      jz @F                          ;
      movzx bx,[esi].Obj.Owner       ; owner
      UpdateWin Trains,bx            ; update trains window
      call UpdateEntireTrain         ;
@@:   call CalcTrainAcceleration     ;
      mov bx,[esi].Obj.Num           ;
      UpdateWin VehView,bx,TVIcons.StopGo ; update order on train view
window
      ret                            ;
mt_notstation:
      push bx                        ;
       cmp [esi].Obj.Action,Order.LeavingStation ;
       jne @F                        ;
       mov [esi].Obj.Action,Order.Null ;
       push ax                       ;
        mov bx,[esi].Obj.Num         ;
        UpdateWin VehView,bx,TVIcons.StopGo ; update order on train view
window
       pop ax                        ;
@@:   pop bx                         ;
      jmp mt_updatepos               ;
```

```
mt_newblock:
      call CalcDirectionToNewBlock  ; get ebp=Direction to new block
      movzx edi,di                  ;
      call GetPossibleRoutes        ; get dl,dh=routes through new block
      jz mt_endoftrack              ; - reached end of track
      call CheckRailCompany         ; check who owns this block
      jc mt_endoftrack              ; - if not us, then turn around

      mov NextBlockDir,bp           ;
      call SetRouteAndPosition      ; get new route and calc position
      jc mt_trackblocked            ; - track blocked (red signal etc)
mt_signaloverride:
      call ObjectBlockInteraction   ; interact with block at proposed
new position
      or ebp,ebp                    ; see if move forbidden
      js mt_endoftrack              ;
      cmp [esi].Obj.LocalType,TrainType.Locomotive ;
      jne @F                        ;
      mov [esi].Obj.Halted,0        ; reset waiting count
@@:   test ebp,bit30                ; see if already set
BlockAddress,TrainTrackType
      jnz @F                        ;
      mov [esi].Obj.BlockAddress,bp ; set new block address
      mov [esi].Obj.TrainTrackType,0      ;
      bts WORD PTR [esi].Obj.TrainTrackType,di ; calc TrackType byte
@@:   push ax                       ;
       mov ax,OldBlockAddress       ;
       call CheckSignals            ;
      pop ax                        ;
      call SlowForCurves            ; slow down train around corners
      mov [esi].Obj.Dir,dl          ; set new direction

mt_updatepos:
      call CalcVisualDirection      ; calculate visual direction
      push bx                       ;
       call SetObjectSize           ;
       call SetGraphicID            ; select new graphic
      pop bx                        ;
      call SetNewPosition           ; set new coords, update on screen
      cmp ObjectTower,0             ; see if loco
      jne mt_dotrailer              ;
      call HillSpeed                ; adjust speed for hills
      call CollisionCheck           ; check for collision with other
trains
mt_dotrailer:
      mov di,[esi].Obj.Trailer      ; get trailer
      cmp di,-1                     ; -1=no trailer
      je mt_exit                    ;
      mov ObjectTower,esi           ; save tower's address
      GetObjAddr esi,di             ;
      jmp mt_trailer                ; move trailer

mt_intunnel:
      call CalcNewPos               ; calc new coords, block address
      mov dl,fs:[di]                ; see if on tunnel block
      and dl,11110000b              ;
      cmp dl,Mno.CivEng shl 1       ;
      jne mt_stillintunnel          ;
      test BYTE PTR gs:[di],11110000b ;
      jnz mt_stillintunnel          ;
      mov bp,di                     ; block address
      call ObjectBlockInteraction   ; see whether exiting tunnel yet
      test ebp,bit30                ;
```

```
      jnz mt_updatepos              ; - out of tunnel
mt_stillintunnel:
      mov [esi].Obj.L,ax            ; update coords
      mov [esi].Obj.R,cx            ; (except U)
      call CalcObjectArea           ;
      jmp mt_dotrailer              ;

mt_endoftrack:
      cmp ObjectTower,0             ; train has reached end of track
      je mt_reverse                 ; if loco, then just reverse train

; disconnect trailer
      mov [esi].Obj.LocalType,TrainType.TrailerStatic
      mov edi,ObjectTower
      mov [edi].Obj.Trailer,-1
      jmp mt_exit

mt_trackblocked:
; train is at red signal
; check length of time waiting for track to clear, and reverse train
; if too long
      cmp [esi].Obj.TrainPassSignal,0 ; see if signal override on
      jne mt_signaloverride         ;

; check for waiting at single-direction signal
      mpush esi,ebp                 ;
       movzx edi,bp                 ; signal block
       mov ax,RouteType.Rail        ;
       call GetRoutes               ;
      mpop ebp,esi                  ;
      movzx edx,NextBlockDir        ; direction onto signal block
      and ax,LegalRoutes2[edx-1]    ; valid routes through block
      jz @F                         ;
      bsf dx,ax                     ; get edx=TrackType
      mov al,SignalBitsRev[edx]     ;
      test al,BYTE PTR LData5[edi*WORD] ; see if mono or bi-directional
      jnz @F                        ;
      mov [esi].Obj.Speed,0         ;
      mov [esi].Obj.SpeedFraction,0 ;
      mov [esi].Obj.Delay,100       ;
      inc [esi].Obj.Halted          ; count no. game cycles stopped for
      cmp [esi].Obj.Halted,254      ;
      jae mt_reverse                ;
      jmp mt_exit                   ; don't reverse if mono-directional
@@:
      mov al,SignalBitsNorm[edx]    ;
      test al,BYTE PTR LData5[edi*WORD] ; reverse if no signal this
direction
      jz mt_reverse                 ;
@@:

; see if time to turn around
      mov [esi].Obj.Speed,0         ;
      mov [esi].Obj.SpeedFraction,0 ;
      mov [esi].Obj.Delay,10        ; force move attempt very soon
      inc [esi].Obj.Halted          ; count no. game cycles stopped for
      cmp [esi].Obj.Halted,254      ;
      jae mt_reverse                ;

; check for train waiting on other side of signals, and immediately
reverse
; if found one
      mpush esi,ebp                 ;
       movzx edi,bp                 ; signal block
```

```
        mov ax,RouteType.Rail          ;
        call GetRoutes                 ;
       mpop ebp,esi                    ;
       movzx edx,NextBlockDir          ; direction onto signal block
       and ax,LegalRoutes2[edx-1]      ; valid routes through block
       jz mt_reverse                   ;
       bsf ax,ax                       ; get TrackType
       movzx eax,ax                    ;
       movzx edx,DirForRoutes[eax]     ; direction beyond signal block
       add bp,BlockIncs[edx-1]         ; block beyond signals
       xor dl,4                        ;
       mov TrainSearchBlock,bp         ;
       mov TrainSearchDir,dl           ;
       mov TrainSearchFlag,0           ;
       push esi                        ;
        mov di,bp                      ;
        mov eax,OFFSET TrainSearch1    ;
        call CallForNearObj            ; search for train other side of
signals
       pop esi                         ;
       cmp TrainSearchFlag,0           ;
       je mt_exit                      ;
mt_reverse:
       mov [esi].Obj.Halted,0          ; reset waiting counter
       mov [esi].Obj.Speed,0           ;
       mov [esi].Obj.SpeedFraction,0   ;
       call ReverseTrain               ; reverse entire train
mt_exit:
       ret                             ;
MoveTrain ENDP
```