Spriten up your Application

PxPlus 2019 (v16)

Goal of this Presentation

How to use Sprites to provide a more Visual interface to your business application

What's a Sprite?

Definition

What's a Sprite?

Definition

- From Technopedia:

> A sprite is defined as a two-dimensional image or animated image that plays a specific role, often independently manipulated, within a larger image environment.

Goal of this Presentation

- Learn how to use Sprite buttons
  - Provide feedback on current state of operations
    - Table use in Restaurant
    - Parking lot spots
    - Hotel/Campsite usage
  - Simplify user interface
    - Point and click/touch visual representing item of interest
    - Drag/move items to indicate change of status
    - Better usability across devices such as Phones and Tablets

Agenda

- Sample applications
  - Restaurant Layout - *demo/2019/tables
  - Parking Lot - *demo/2019/carlot
- The logic behind the scenes
  - What you will need
  - Dealing with scaling the images
  - Handling image rotation
  - Moving the images around
- Other uses for Sprites

Sample Applications

- Previewed in mailing leading up to DireXions
- Two applications for use of Sprites

Restaurant Layout

Parking Lot

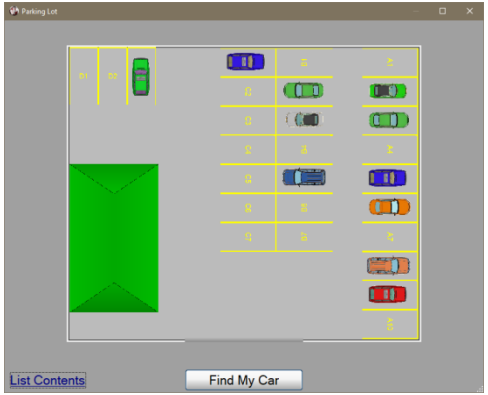


Table Layout Application

BEHIND THE SCENES

Getting Started

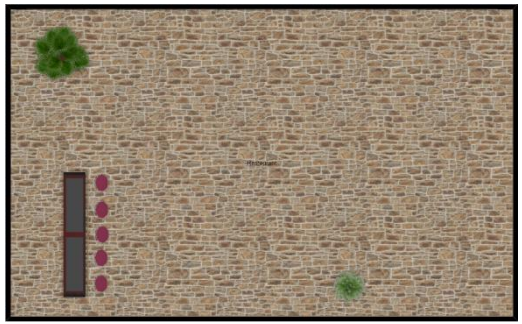
What you will need

- Images
  - Backdrop image
    - The image that defines the scene or location
      - Building Floor plan(s)
      - Parking Lot
      - Warehouse layout
  - Sprite images
    - The objects the user will handle (tables, cars, etc.)
    - Images should be PNG or GIF with transparent background
      - Animated GIFs can also be used
  - You will need the real ACTUAL dimensions of the background and objects
    - Size should represent the area the image covers

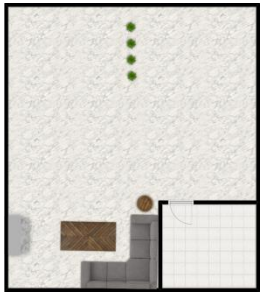
Getting Started

Floorplans and Tables

Floor 1
50 x 31 ft



Floor 2
32.5 x 36.5 ft



Tables

7 x 4.5 ft

5.5 x 3.3 ft

5.5 x 5.5 ft

7.5 x 7.5 ft

7.75 x 7.75 ft

7.75 x 5.5 ft

Data Files

Image Definition Files

- Rooms

RoomName	Wide	High	Image
Dining Room	50	31	Floor1_50x30.png
Lounge	32.5	36.5	Floor2_32.5x36.5.png

- Tables

TableType	Wide	High	Image
2ftRnd-2	7	4.5	2ft_rnd_2ch.png
3ftSqr-2	5.5	3.33	3ft_sqr_2ch.png
3ftSqr-4	5.5	5.5	3ft_sqr_4ch.png
5ftRnd-6	7.5	7.5	5ft_rnd_6ch.png
5ftSqr-8	7.75	7.75	5ft_sqr_8ch.png
5x3Rect-6	7.75	5.5	5x3ft_rect_6ch.png

Data Files

Current Layout

- Layout

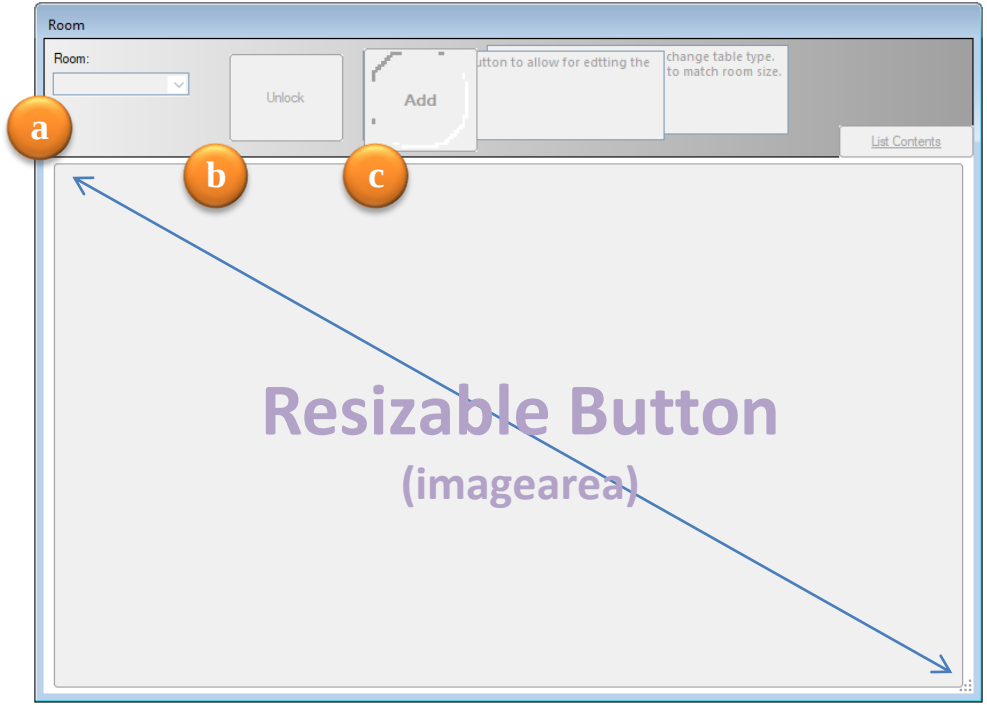
RoomName	TableName	TableType	Left	Down	Angle	State	ChangedTime*
Dining Room	Table 1	2ftRnd-2	20	10	0	0	-4
Dining Room	Table 2	2ftRnd-2	30	10	0	1	-10
Dining Room	Table 3	2ftRnd-2	45	25	0	0	-30
Dining Room	Table 4	5x3Rect-6	20	20	0	1	-11
Lounge	Table 11	5x3Rect-6	8	5	0	1	-1
Lounge	Table 12	5x3Rect-6	8	14	0	1	-2
Lounge	Table 13	5x3Rect-6	8	23	0	1	-2
Lounge	Table 14	5x3Rect-6	24	5	0	1	-3
Lounge	Table 15	5x3Rect-6	24	14	0	1	-3

* This normally would have actual time of last update but for purposes of demo it only has a relative time in terms of minutes.

Nomads Panel

- Actual background occupied by resizable hidden button
  - Button size determines region to show background
  - Panel set to custom resizing
    - Button was only Auto-size

- Other controls
  a. Drop box to set Room$
  b. Check box lock/unlock
  c. Add button



Drawing the Room

```
Draw_Room:
!
 precision 6
 gosub Clear_room ! Remove any ex
!
 image_top = int(IMAGEAREA.CTL'top)
 image_left = int(IMAGEAREA.CTL'left)
 image_height = int(IMAGEAREA.CTL'height)
 image_width = int(IMAGEAREA.CTL'width)
!
 pixel_per_line=image_top/IMAGEAREA.CTL'line
 pixel_per_col=image_left/IMAGEAREA.CTL'col
!
 read (rooms,key=room$)
 cur_room$=roomName$
!
 pixels_per_foot= min(image_width/wide,image_height/high)
!
 room_width=int(pixels_per_foot*wide)
 room_height=int(pixels_per_foot*high)
```

Use Pixel values from resizable button

Determine pixels per foot

Determining

Center room in imagearea

```
!
 room_top=int((image_height-room_height)/2)+image_top
 room_left=int((image_width-room_width)/2)+image_left
!
 print 'image'(delete "ROOM"),'image'("ROOM"),
    'picture'(@x(=room_left),
   @y(=room_top),
   @x(=room_left+room_width),
   @y(=room_top+room_height),
    image$,6),
!
! Now draw the tables
!
 select * from Layouts begin cur_room$ end cur_room$+$FE$
 tbl=++max_table_ctl
 gosub Draw_table
 next record
!
 cur_size$=IMAGEAREA.CTL'width.height.$  ! Save current size
 return
```

Delete old image and draw

Trigger to redraw

Drawing a Table

```
!
! Draw_Table  Draw current table using CTL in 'tbl'
!
Draw_table:
 read (Tables,key=TableType$,dom=*
!
 if angle \
  then gosub Angle_Adjustment ! adjust size based on angle
!
 item_width=int(pixels_per_foot*wide)
 item_height=int(pixels_per_foot*high)
!
 item_left=
    int(room_left+left*pixels_per_foot-(item_width/2))
 item_top=
    int(room_top+down*pixels_per_foot-(item_height/2))
!
 item_col=max(0,(item_left-2)/pixel_per_col)
 item_cols=(item_width+4)/pixel_per_col
 item_line=max(0,(item_top-2)/pixel_per_line)
 item_lines=(item_height+4)/pixel_per_line
!
 X$=image$
 if angle<>0 then X$+=",r:"+str(angle)
!
 button tbl,@(item_col,item_line,item_cols,item_lines)=
    "{"+X$+"}"+TableName$,opt="~FTC",
    fnt=",-"+str(1.5*pixels_per_foot)+",B"
!
 tbl'textcolor$=tbl(state,"Green","Yellow","White","Red")
 tbl'Moveable=not(locked)
 return
```

If on an angle, compute true size

Compute size and position based on Pixels/feet

Set moveable, if screen not locked

Dealing with Angles

Geometry 101

```
!
! Handle rotation which makes button size vary
!
Angle_adjustment:
!
 Radians=(angle*3.14159265359)/180
 Sine=abs(sin(Radians))
 Cosine=abs(cos(Radians))
!
 W= (wide*Cosine) + (high*Sine)
 H= (wide*Sine) + (high*Cosine)
!
 wide=W
 high=H
!
 return
```

Rotating creates different image dimensions

Moving the Tables

Handling Events

- Each table is really a button and fires unique CTL (100-355)
  - Use %Nomads'Fkey_Handler$ to trap/process

```
INITIALIZE_DATA:
 sv_fkey_handler$=%NOMADS'Fkey_Handler$
 %NOMADS'Fkey_Handler$=pgn+";fkey_Handler"
```

```
WRAPUP:
 %NOMADS'Fkey_Handler$=sv_fkey_handler$
 return
```

  - When table generates CTL, program determines Table and processes event

```
FKEY_HANDLER:
  if ctl>=100 and ctl<355  then
     tbl=ctl;
     TableName$=arg(tbl'text$,2,"}");
     gosub SELECT_STATE
!
 return
```

Get table name from button text

```
SELECT_STATE:
  read (Layouts,key=room$:TableName$,dom=*return)
!
  button read tbl,_eom$
  if _eom$="M" then
     gosub Table_moved;
     return
!
! … popup menu  logic…
```

Moving the Tables

```
Table_moved:
!
 leftPixel = tbl'left
 topPixel = tbl'top
!
 precision 10
!
 read (Tables,key=TableType$,dom=*return)
!
 if angle then \
    gosub Angle_adjustment ! Adjust size based on angle
!
 item_width = int(pixels_per_foot*wide)
 item_height = int(pixels_per_foot*high)
!
 xPixel = leftPixel - room_left + item_width/2
 yPixel = topPixel - room_top + item_height/2
!
 left = xPixel / pixels_per_foot
 down = yPixel / pixels_per_foot
!
 precision 2
!
 left = int(left*2)/2 ! Snap to .5 of a foot
 down = int(down*2)/2
!
 roomName$ = room$
 write (Layouts) ! Update layout
!
 read (Layouts,key=CUR_ROOM$:TableName$)
 button remove tbl
 gosub Draw_table ! Redraw the table
!
 return
```

Get new table (button) position

Compute center of table

Snap movement to specific increments

Update layout file and redraw table

The Car Park

- Similar to restaurant
  - Add moving logic to determine parking spot
    - Uses file containing location/size of each parking spot
    - Size of Sprite based on spot size (not car size)
    - Rotation is automatic to force cars with 'front' in
      - Each slot has angle to rotate car
- Could be expanded for car dealership or rental business

Other Ideas

- Warehouse
  - Show where products are kept to assist in locating / fulfillment
  - Likely would not scale image
    - Small products could result in tiny unusable images

- Hotel, campground or other facility
  - Show rooms/locations in use
  - Show state of room (occupied, cleaned, etc.)
  - For security, could show picture of resident (e.g. retirement home, cruise ship)
    - Provide click on picture for more information

Other Ideas

- Dynamic POS terminal
  - Allow user to layout the POS workstation buttons
  - Individual buttons for each item
- Dynamic Menu system
  - Allow users to layout out their menu to suit their needs
- Work site
  - Show location of crew and equipment

Recap

- Reviewed what is needed
  - Images and Background
  - Sizes and position of information

- *dummy* Nomads control used to handle sizing

- Discussed key items
  - Handling dynamic sizing of items based on screen
  - Dealing with the impact of rotation
  - Moving sprites

There are many options.

All you need is a little imagination!

Additional Resources

The help link(s) below refer to the current on-line help pages. The functionality may have been further updated since the PxPlus 2019 (version 16) release.

- Buttons – Bitmap placement/moveable options – Directive
- Buttons – Bitmap placement/moveable options - NOMADS
- 'Image' Mnemonic (Define a graphics group)
- NOMADS Fkeys