CS250 Intro to CS II

Spring 2013

Dark GDK
Sprites

- Up to now, you've written console apps and Dark GDK apps that draw primitive shapes
- It's time to use images that have been created with a software app (e.g. Paint, Photoshop) or captured with a digital device (e.g. scanner, camera)

Images

- Images are commonly saved as bitmaps
- Dark GDK provides functions for loading, displaying, and modifying bitmaps
- bitmap – data that describes every pixel in an image
- Dark GDK has a function dbLoadBitmap that loads a bitmap file into memory
- Acceptable file formats are: .bmp, .jpg, .tga, .dds, .dib, or .png

Sample Dark GDK Bitmap Program

```
#include "DarkGDK.h"

void DarkGDK ()
{
  // Load and display image
  dbLoadBitmap ("sprite.png");

  dbWaitKey ();
}
```

Where to place Bitmaps?

- Consider a Studio solution called CS250 with a project called Sprites.

- Bitmaps are placed in the Sprites folder unless otherwise specified.

Dark GDK
Sprites

- A sprite is a graphic image used in serious game development.
- Dark GDK provides many useful functions to manipulate sprites.
- The function dbLoadImage loads images into memory
- The function dbSprite creates a sprite from the images loaded into memory

dbSprite

- The general format for creating a sprite is dbSprite (spriteID, x, y, imageID) where
  - spriteID is in the range of 1 to 65535 inclusive
  - x is the x-coordinate of the upper-left corner of the image
  - y is the y-coordinate of the upper-left corner of the image
  - imageID is the number of one of the images loaded into memory

Simple Sprite Program

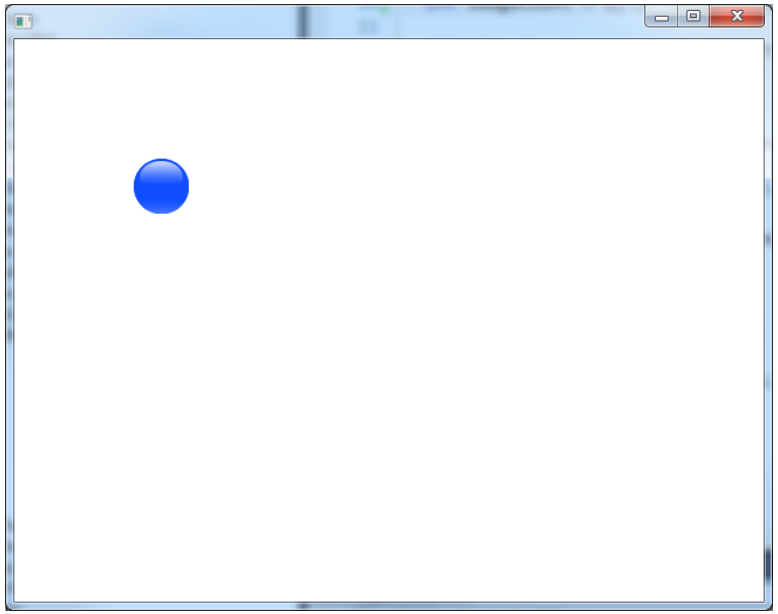
```
#include "DarkGDK.h"
#include "Sprite.h"

void DarkGDK (void)
{
  dbLoadImage ("Sprites/ball_blue.png", 1);

  dbSyncOn ();
  dbSyncRate (60);

  while (LoopGDK ())
  {
    dbClear (255, 255, 255);
    dbSprite (1, 100, 100, 1);

    dbSync ();
  }
}
```



Dark GDK
Sprites

```
#include "DarkGDK.h"
#include "Sprite.h"
#include "Image.h"


void DarkGDK (void)
{
  int screenWidth = dbScreenWidth (),
      screenHeight = dbScreenHeight (),
      whichSprite;

  dbRandomize (dbTimer ());
  Image::loadImages ("sprites");
  whichSprite = 1 +
     dbRND (Image::getNumberOfImages () - 1);
  dbSyncOn ();
  dbSyncRate (60);

  while (LoopGDK ())
  {
    dbClear (255, 255, 255);
    dbSprite (whichSprite, screenWidth / 2,
              screenHeight / 2, whichSprite);

    dbSync ();
  }

}
```

Problem #1

- Modify the SpriteExample as follows:
  1. Create an array of pointers to Sprites
  2. Dynamically allocate space for 25 balls. Make sure the balls are placed somewhere on the screen. The dimensions of the Sprites are 50x50.
  3. Display the Sprites on the screen.
  4. When the program is terminated, free all dynamically allocated Sprites.
  5. Using the debugger, check to see that all space is freed.

Problem #2

- Load up the deck of cards in the Cards folder
- Display one card every second