Game Programming with DXFramework

Jonathan Voigt
voigtjr@gmail.com
University of Michigan
Fall 2006
The Big Picture

- DirectX is a general hardware interface API
- Goal: Unified interface for different hardware
- Much better than the past
  - Programs had to be coded for specific hardware
DXFramework is a Simple DirectX Game Engine

DXFramework goals:
- Simplicity
- 2D support
- Object oriented design
- Instruction by example
Types of Games to Create

Simple!
Fun!
Easy!
(2D!)
Iterative Development

Design → Implementation → Evaluation → Analysis
Student Games

Only the final projects are available on the web (not the arcade games)

Fall 2004 (DXFramework 0.9.3):
http://ai.eecs.umich.edu/soar/Classes/494/showcase-2004/Games.htm

Fall 2005 (DXFramework 0.9.6):
Arcade Game Demo
DXF Capabilities

- Genres: arcade, action, puzzle, role playing, adventure, strategy
  - Top down, side view, isometric
- Many other possibilities!
DXF Capabilities

• Sounds & Music
  – Midi background, sound effects
  – simple pan & volume control

• Input
  – Keyboard and mouse
  – Joystick possible: use USB joystick and be prepared to turn it in with your game!
DXF and DXUT

• Microsoft’s DirectX utility library
  – Included with DirectX SDK
• Included with DXFramework
  – ‘dxut’ project
• See DirectX samples for more on DXUT and DirectX
DXF Prerequisites

- Windows 2000/XP
- Microsoft Visual Studio 2005
- Latest DirectX SDK
- Windows SDK
- Python interpreter
- Creativity
Installation

• Refer to Getting Started guide:

• Generally speaking:
  – Install Visual Studio & SDKs
  – Configure Visual Studio
  – Download and Extract package
DXFramework Concepts
A DXF Application is a graph of Game States

- You create your game by defining game states (extending a GameState class) and the conditions for transitioning between them.
Tetris as a graph of states

Start

Title

Options

Game

High Score

Entered Name

Select Players

Type & Music

High Score

Quit

Not a high score
Global Data
(data shared across states)

• What about global data?
  – High scores
  – Option settings

• Store global data in the Registrar
  – The registrar is part of your project

Registrar
(public dxf::RegistrarInterface)
RegisterStates()
 Initialization

The first state registered is used as the initial state!
Execution

• The next thing main() does is call Run()
  – This starts the main loop: 
    Input ➔ Update ➔ Render
  – Each iteration of this loop represents a frame

• This loop executes as fast as possible
  – DXF uses variable discrete
  – Faster hardware runs faster
  – Time elapsed is available to Update()

• When Run() exits, so does the program
Key Points in the Game Loop

- Load()
- Update()
- Render2D()
- DXFChangeState()
- Unload()
Creating States

• Extend dxf::GameState2D
  – Implement the necessary functions
• Need a complex GUI?
  – Extend dxf::GameStateGUI instead
• Need sub-states?
  – Advanced topic
  – Extend dxf::StateManager as well
Registering States

- Registrar
  - RegisterStates()
  - DXFRegisterState(string, state pointer)

const std::wstring Registrar::kTitle = L"Title";
const std::wstring Registrar::kKeyboard = L"Keyboard";
...
dxf::DXFRegisterState(kTitle, Title::Instance());
dxf::DXFRegisterState(kKeyboard, Keyboard::Instance());
...
dxf::DXFChangeState(Registrar::kKeyboard);
DXF Engine Architecture

dxf::Game

*top level container*

- dxf::Model
  - states

- dxf::Console
  - debugging

- dxf::View
  - video device

- dxf::Controller
  - input devices
Other DXF Components

- **Sprites**
  - Almost everything on the screen
  - Many acceptable formats (like .jpg, .png)
- **Sounds**
- **Fonts**
- **Console**

- All usually members of game states or registrar
Sprites are Everywhere!

Origin: (0, 0)

Back buffer/screen: 800x600

Sprites

Position: 782, 462

Width: 16 pixels

Height: 64 pixels

Paddle Sprite

Zeros (text) are not sprites, they are created by special Font object
The Back Buffer

- Sprite ‘cache’ or ‘canvas’
- Same size as screen when full-screen
- Size of window ‘client area’ when windowed
Drawing to the Back Buffer (Render2D)

```cpp
Pong::Render2D() {
    center.Render2D();
    scoreboard.Render2D();
    font.Render2D(...);
    font.Render2D(...);
    left.Render2D();
    right.Render2D();
    ...
    ball.SetAnimation(1);
    ball.SetColor(...);
    ball.Render2D(...);
    ball.SetColor(...);
    ball.Render2D(...);
    ball.SetColor(...);
    ball.Render2D(...);
    ...
    ball.Render2D();
}

Title::Load() {
    DXFSetClear(true);
    DXFSetClearColor(WHITE);
}
```
Button Input

Button is Up  Button is Down

BUTTON_UP  BUTTON_PRESSED

BUTTON_RELEASED  BUTTON_DOWN

DXFCheckKeyboard()  DXFCheckMouse()  DXFCheckJoystick()
Mouse Input

- DXFGetMousePosition()
  - Returns X,Y position on back buffer
- Passing this to Sprite's CheckIntersection function is useful
  - See Button in DXFramework-Demo
  - Very recent bug fix, see discussion or FAQ for details, or download a new copy of the framework
Collision Detection

- Simple: Check bounding rectangles

- No collision
- Overlap in y
- Overlap in x
- Overlap in both dimensions (Collision)
Collision Detection

- Simple: Check bounding circles
  - Distance between center points
  - Collision if distance between center points is less than sum of radii
Fonts

- Use the font class to draw text to screen
- Text is expensive
  - Keep amount of text low
- Consider text rendered on sprites
Sounds

- Use sound class for sounds
- Wave files, Midi files, MP3, others
  - Ogg? Not sure
- Usage similar to sprites
  - Create using filename
  - ‘Render’ using Play
The DXF Console

- Essential debugging tool
  - No stdout available!
  - A decent substitution
- ` ` key toggles
- Output using Console::output like you would use cout:
  - Console::output << "The number is: " << x << std::endl;
- Output is flushed only when a newline is encountered!
Creating and Registering Custom Commands

- Registrar’s other function registers custom console commands
- Define command in global scope with correct function signature
- Pass pointer and string to DXFRegisterCommand
Using the DXUT GUI with DXFramework states

- Program by example
- See comments in UI Demo
Questions? Need help?

- I’m here to help
- Check the FAQ on the Wiki
  - I’ll fill in content as I get it
- Post in the discussion forum
- Send me mail to schedule an appointment
  - voigtjr@gmail.com
  - 3828 CSE Building