717310: Game Programming

Steffan Hooper

Thursday, 6 August 2015

Overview

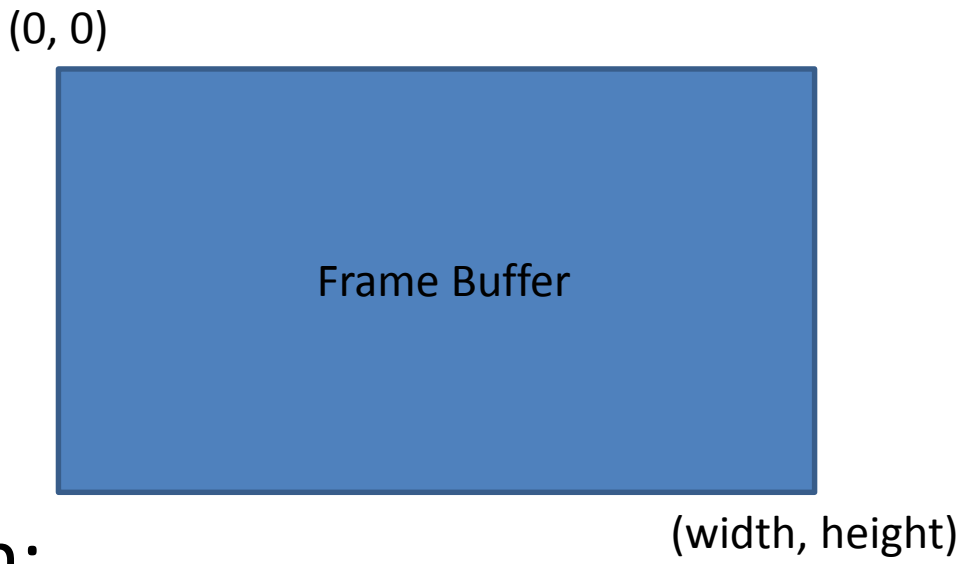
- Graphical Effects for Computer Games
- 2D Graphics
  - Frame Buffer, Sprites, Pixel Art
  - User Interface, Heads-up-display
- 2D Game Framework
  - Roll-our-own C++ 2D Graphics Framework
  - Sprites, Input, Entities, Collisions, Debugging…
- Exercises

Graphical Effects for Computer Games

- Computer Graphics
  - Pixel: Picture Element
    - The smallest addressable visual element.
  - Colour Format:
    - RGB: R8G8B8: 24 bit colour: 8 bits red, 8 bits green, 8 bits blue.
  - The Frame Buffer: Array of pixels in RAM.
    - Size = (Width * Height * Colour Depth) bytes…
  - Aspect Ratio: Width:Height or Width/Height
  - Refresh Rate: 60Hz, 100Hz, etc…

Graphical Effects for Computer Games

- The Frame Buffer's Coordinates



- Resolution:
  - Is: (width * height * colour depth)
  - Eg: 800px * 600px * 24bits = 1.44 Megabytes

Graphical Effects for Computer Games

- 2D Graphics: Pixel Art
  - Digital art edited on a pixel level…
  - Techniques:
    - Line art, limited palette, dithering, hand-made anti-aliasing…
  - Types:
    - Isometric, Non-isometric
    - Game Examples:
      - Sim City 2000 (Maxis, 1994)

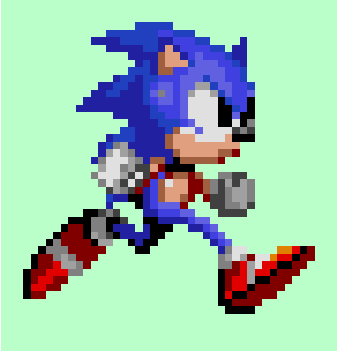
http://piq.codeus.net/static/media/userpics/piq_30130_400x400.png

http://i245.photobucket.com/albums/gg62/witchyhoy3/2479.gif

Graphical Effects for Computer Games

- 2D Graphics: Sprite
  - Two dimensional image…
  - Generally integrated into a larger scene…

Sonic The Hedgehog 2 (Sega, 1992):

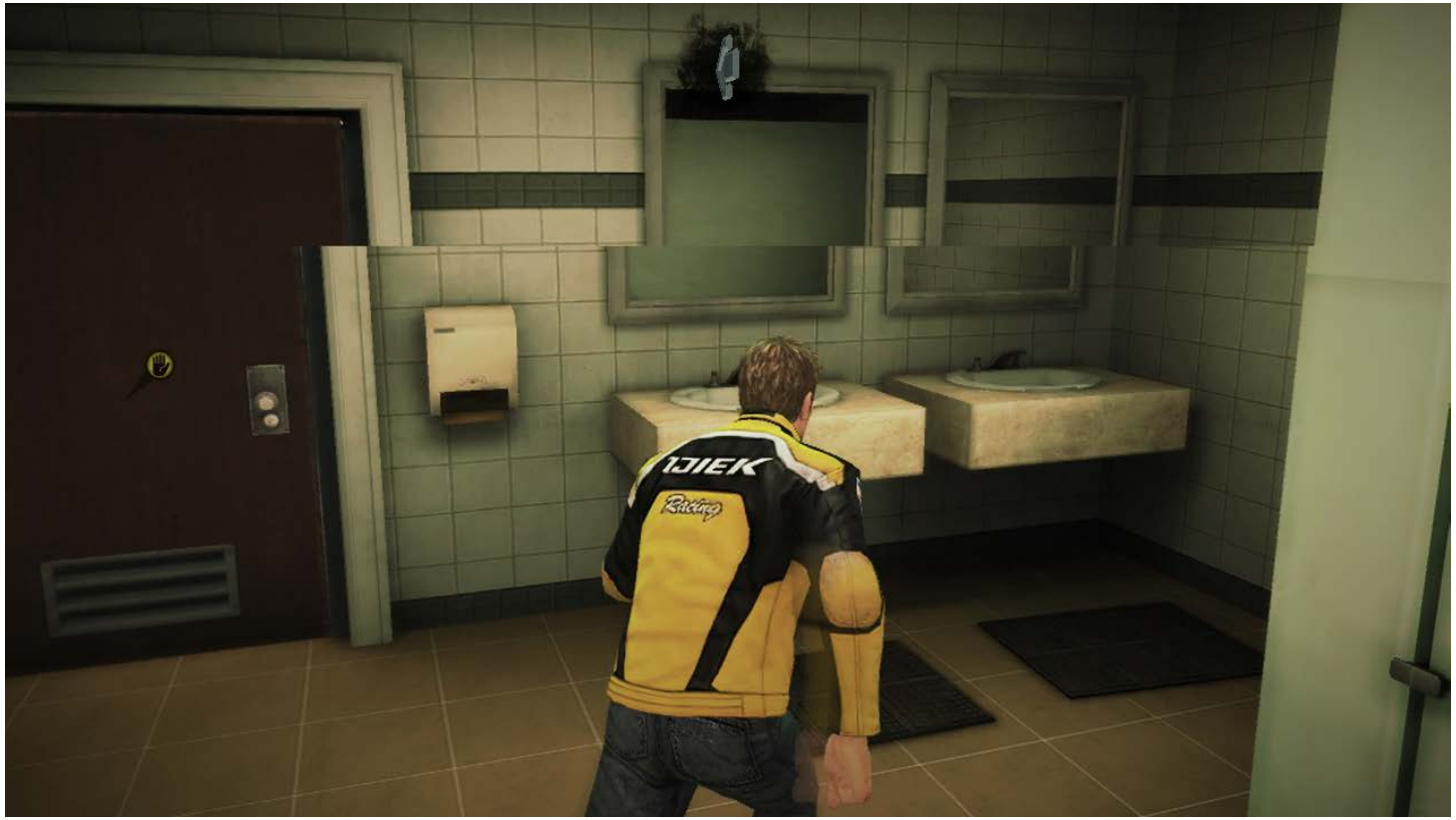


http://www.spriters-resource.com/fullview/10073/

  - Possible Formats: BMP, GIF, PNG, JPG, TGA, …
    - Can be platform specific!

Graphical Effects for Computer Games

- Screen Tearing:



http://zoneitastuces.com/wp-content/uploads/2011/10/Screen-tearing.png

Graphical Effects for Computer Games

- Double Buffering: Two frame buffers!
  - Front: What is currently drawn to the screen.
  - Back: Area for the real-time rendering program to draw to.
  - Once the Back Buffer is completely drawn, it is promoted to become the Front Buffer:
    - Buffer Swapping:
      - Blit: Copying byte for byte, Flip: Pointer swapping.
  - If swapping is synced with screen refreshing:
    - No tearing!

Graphical Effects for Computer Games

- Draw Order: Painters algorithm…
  - For example:
    - `Clear();`
    - `DrawBackground();`
    - `DrawStars();`
    - `DrawEnemies();`
    - `DrawBullets();`
    - `DrawPlayer();`
    - `DrawExplosions();`
    - `Present();`

Graphical Effects for Computer Games

- Frame Counter:

```
void Game::Process(float dt)
{
    m_elapsedMilliseconds += dt;

    if (m_elapsedMilliseconds > 1000)
    {
        m_elapsedMilliseconds -= 1000;
        m_FPS = m_frameCount;
    }
    // Update the game...
}

void Game::Draw()
{
    ++m_frameCount;
    // Draw the game...
}
```

Graphical Effects for Computer Games

- Simple Sprite Movement
  - Per frame… Add or subtract an amount from the position of the sprite… p_x += d_x; p_y += d_y;
- Mathematics: Point rotation
  - Rotate a point (p_x, p_y) around a point (c_x, c_y):
    - Translate point back to the origin…
    - Rotate point around the origin…
    - Translate rotated point back.
    - $r_x = \cos(angle) * (p_x - c_x) - \sin(angle) * (p_y - c_y) + c_x$
    - $r_y = \sin(angle) * (p_x - c_x) - \cos(angle) * (p_y - c_y) + c_y$

Graphical Effects for Computer Games

- So far…
  - These sprites have been raster generated…
    - Created pixel by pixel… possibly hand drawn first!
- Other Ways to Create Sprites:
  - Vector-based: Good for scaling…
  - 3D-based: Pre-rendered…
    - Examples:
      - Donkey Kong Country (Rare, 1994) on the SNES.
      - Killer Instinct (Rare, 1994) on the SNES.
      - Diablo (Blizzard North, 1996) on Windows.

Graphical Effects for Computer Games

- Alpha Transparency
  - See through part of the sprite…
    - Layering of sprites created by draw order…
  - Use PNGs…
    - Beware: BMP does not have alpha!
  - Mask Layer…
    - Adobe Photoshop
    - Generally signalled by a grey and white checkerboard…

Graphical Effects for Computer Games

- 2D Graphics: User Interface (UI)
  - Menu, Widgets: Buttons, Toggles, Sliders, etc…

http://www.riaxe.com/wp-content/uploads/2013/10/game-ui.jpg

https://lh3.ggpht.com/pNlAVrlfHmXIC3NDGeR7gQwwXcjgGi-2t_3xw8Z9jRfrrMsDqFILTTeN_hJo1PTVEpI=h900

Graphical Effects for Computer Games

- 2D Graphics: Heads-Up Display (HUD)
  - Status Bar… User Interface…
  - Inform the player visually of current game state…
  - Example:
    - Score
    - Health
    - Level

http://arcadeberg.com/files/2009/04/eso_hud.jpg

Graphical Effects for Computer Games

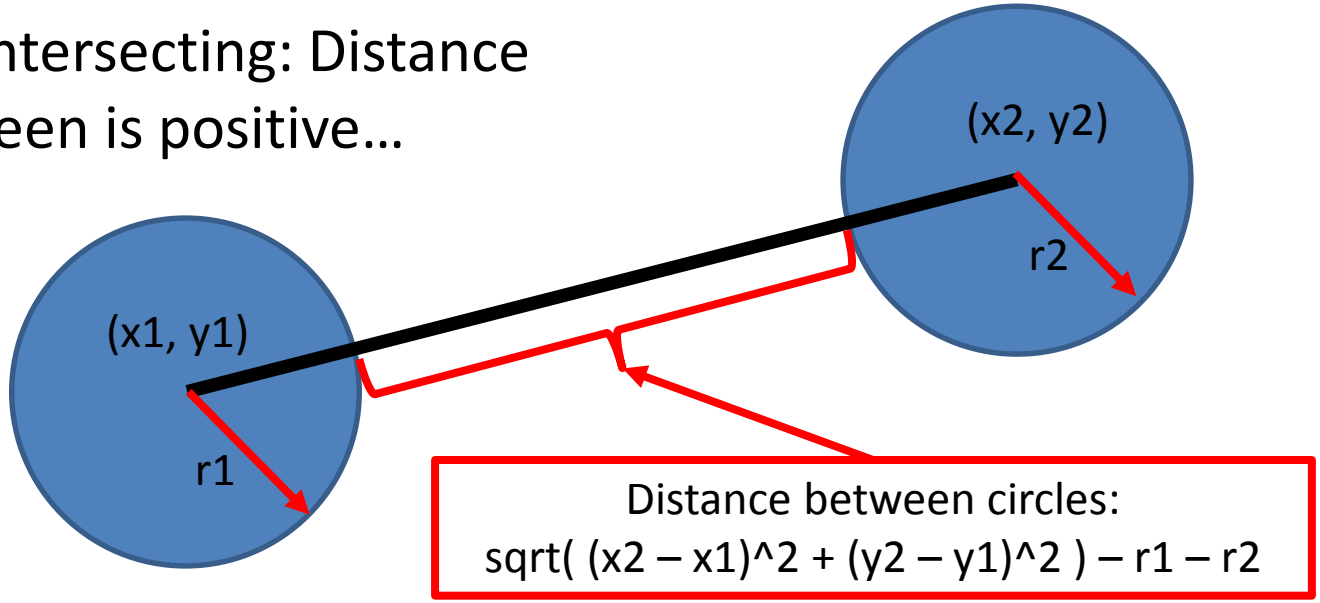
- Collision Detection
  - Basics: Window Boundaries
  - Point vs Point: Pythagorean Theorem.
  - 2D: Bounding Shapes
    - Circle vs Circle
    - Rectangle vs…
    - Line vs…
    - Oriented-Rectangle
  - Shape Grouping: Hierarchies…

Graphical Effects for Computer Games

- Simple Collision Detection
  - Circle vs Circle:
    - Per frame testing for collision:
      - Create two circles: One per object involved in the collision
      - Set the position and radius of each circle…
      - Do Pythagoras…
    - $O(n^2)$ approach: all bullets vs all enemies
      - We will look at improving performance later!

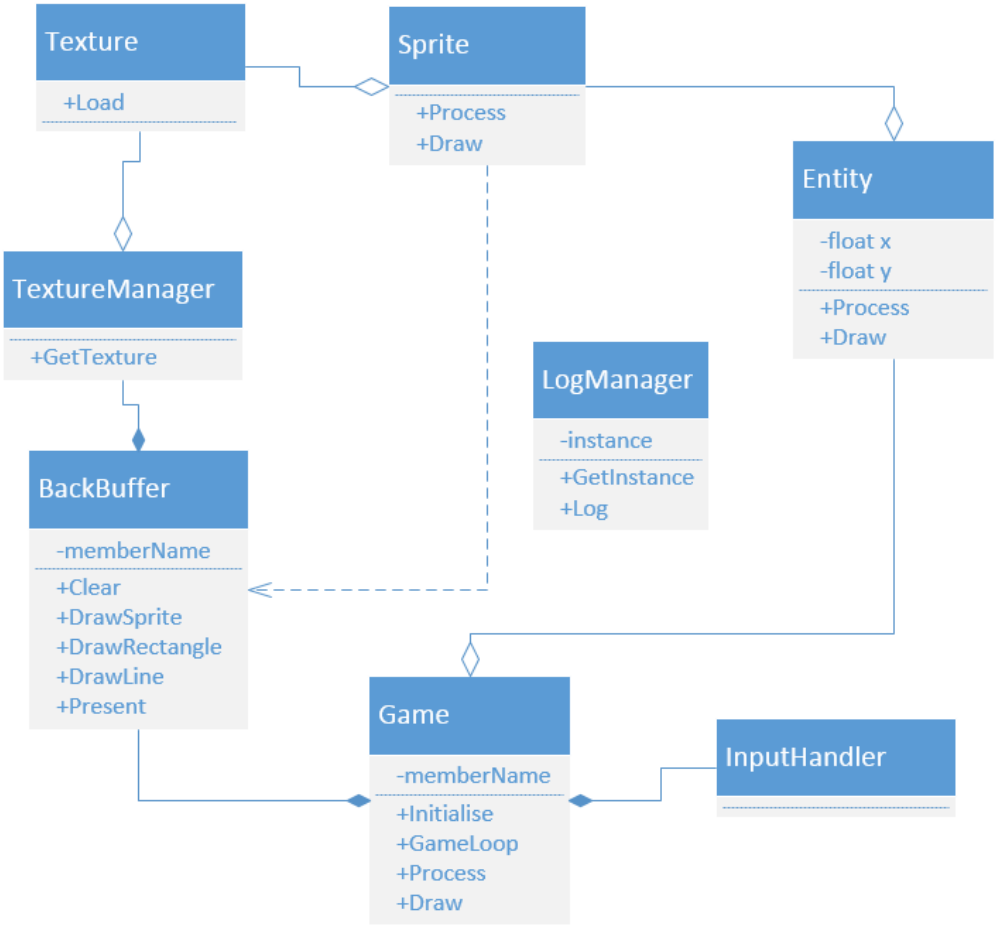
Graphical Effects for Computer Games

- Simple Collision Detection
  - Circle vs Circle:
    - Overlapping: Distance between is negative….
    - Touching: Distance between is 0.
    - Not intersecting: Distance between is positive…



Graphical Effects for Computer Games

- C++ SDL Framework Overview:



Graphical Effects for Computer Games

- **`Sprite`** Class:
  - Responsibilities:
    - Represents a 2D images.
    - Can be drawn to the backbuffer.
    - Has a width and height.
    - Position Control Handle (Top Left/Center/Other?)
    - References a **`Texture`** object…
- **`Texture`** Class:
  - Responsibilities:
    - Loading .png files into memory.

Graphical Effects for Computer Games

- **`TextureManager`** Class:
  - Responsibility: Loading/Storing Textures
    - It is a “Resource Manager”!
  - Member Data:
    - Container of Textures: A HashMap (**`std::map<>`**)
  - Member Functionality:
    - **`Texture* GetTexture(const char* pcFilename)`**
      - Loads a texture from file if not already in the container…
      - Otherwise returns the previously loaded texture from the container…

Graphical Effects for Computer Games

- **`InputHandler`** Class:
  - Responsibility: Receive input, translate input into game events that trigger state changes…
    - Xbox 360 Gamepad!
  - Member Functionality:
    - **`Initialise()`**
      - Sets up the available controllers…
    - **`ProcessInput()`**
      - Controller button goes down…
        - » Receives an event…
        - » Processes the events…

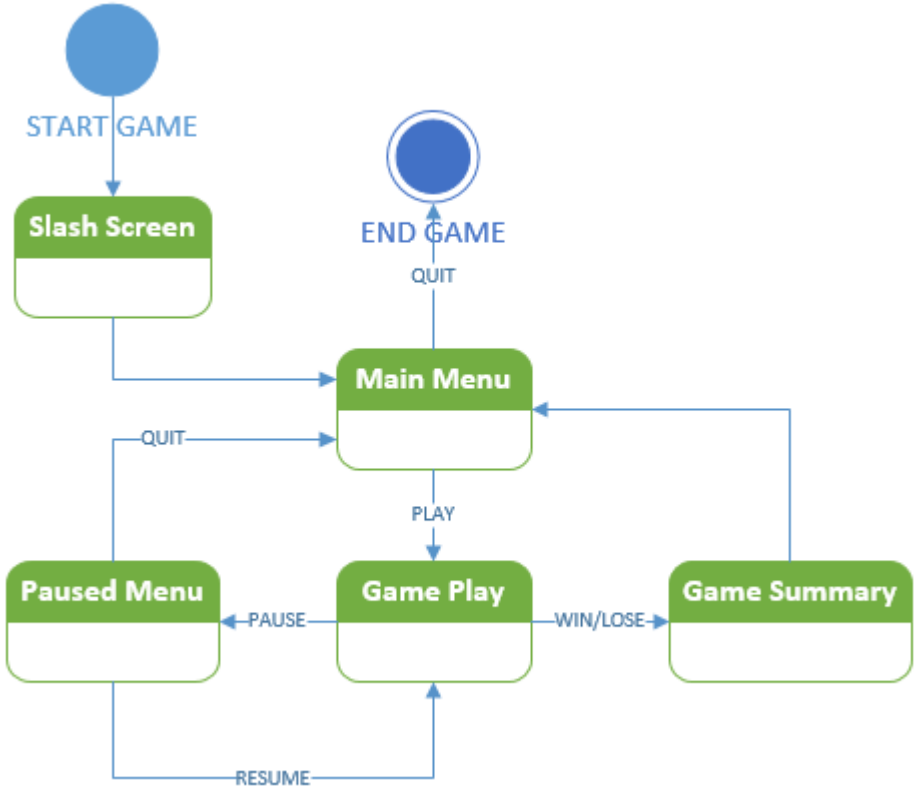
Graphical Effects for Computer Games

- **`Game`** Class:
  - Responsibility: Represents the Game!
  - Member Functionality:
    - **`DoGameLoop()`**
    - **`Process()`**
      - Calculates the time elapsed…
      - Updates game world entities…
    - **`Draw()`**
      - Clears the backbuffer…
      - Game is rendered in this method…
      - Presents the backbuffer…

Graphical Effects for Computer Games

- Simple Game States:
  - C++ example:

```
enum GameStates
{
    SPLASH_SCREEN,
    MAIN_MENU,
    GAME_PLAY,
    PAUSED_MENU,
    GAME_SUMMARY
};
```



Game Entities

- A Game Entity
  - Encapsulates the behaviour of the entity.
  - Abstraction! Derive specialisations…
  - Entities can be updated:
    - Their chance to change their state over time!
      - One frame's worth of behaviour…
- Entity Component System:
  - Software Architecture Pattern
  - Game Entities…

Game Entities

- An **`Entity`** Class

```
class Entity // Pseudo Code!
{
    float x;
    float y;

    Entity() { /* ... */ }

    void process(float dt) { /* ... */ }

    // Accessor Methods for x and y.
}
```

Entity position stored as **`float`**s; converted to **`int`**s for rendering as sprites.

Game Entities

- Entity Component System: Game Entities…
  - Entity examples:
    - Space Invaders (Taito Corporation, 1978):
      - Player, Enemy, Bullet, UFO, Barricade, etc…
  - The **`Game`** Object (or other owner…):
    - Iterates through a container of enemies to process them: update their state…
      - Calling process on each entity in the container!
    - Iterates through a container of enemies to draw…
      - Rendering them to the back buffer!

Game Entities

- Entity Component System: Continued…
  - Game entities can be added or removed during game play…
  - Spawn new enemies in the level:
    - Add to the container!
  - Kill enemies:
    - Remove from the container!
  - Dynamic game behaviour!
    - Add/remove as many enemies as we want…
    - A level designer can do their job!

Game Entities

- The **`Game`** Class:

```
class Game // Pseudo Code!
{
    // Container<Enemy> enemies;

    void process(float dt)
    {
        foreach (Entity e : enemies)
        {
            e.process(dt);
        }
    }
};
```

Game Entities

- Entity Component System: Continued…
  - What about inactive entities?
    - If there is one container:
      - The Entity class will need a Boolean “entity active” flag:
        - » If true, process gets done…
        - » If false, no processing…
    - Or use two containers:
      - Active container and inactive container…
      - Move entities between them as needed…
  - Beware of iterating and changing the container contents while in the iteration!

Graphical Effects for Computer Games

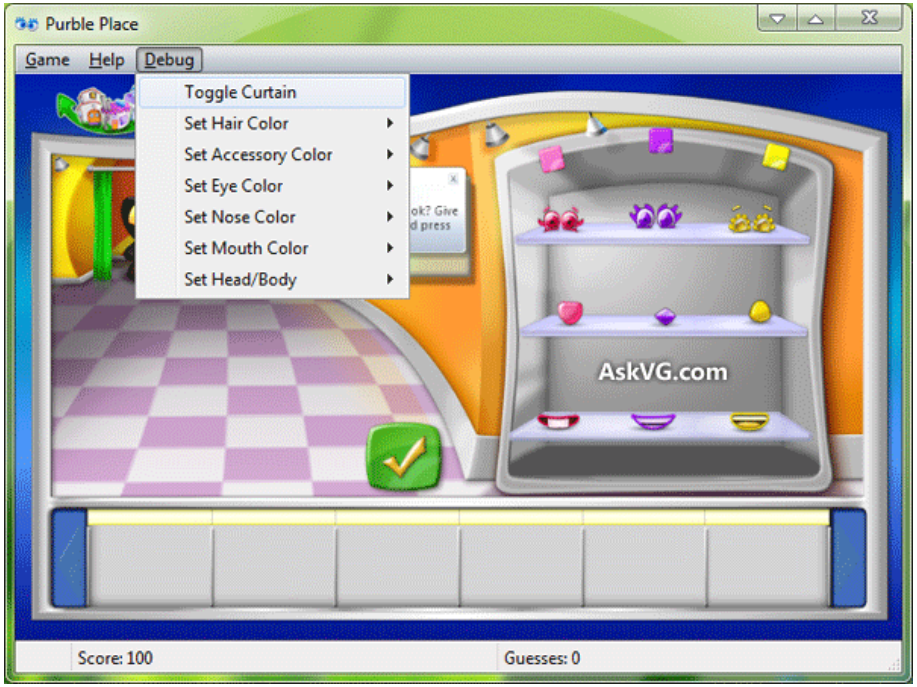
- Simple Enemy States:
  - C++ example:

```
enum EnemyStates
{
    PATROLLING,
    ATTACKING,
    GUARDING,
    SLEEPING,
};
```

  - Design with UML State Diagrams…
    - What states exist for the Space Invaders alien enemies?

Debug Menu / Window

- Debug (Cheats) Menu / Window
  - Hidden…
  - Control game features…
  - Toggle:
    - Player invisibility.
    - Enemy spawning.
    - Level completion.
    - Debug drawing.
    - Any tools/options to help the developer!



http://www.askvg.com/revealing-hidden-secret-debug-menu-to-cheat-in-built-in-microsoft-games-in-windows-7/

Exercises

- Week 3:
  - Framework and Assets: Day 006 Framework.zip

  - Day 006.1 – Space Invaders: UML Design Refined
  - Day 006.2 – Space Invaders: 2D Sprite, Player Ship Movement
  - Day 006.3 – Space Invaders: Alien Enemy Wave
  - Day 006.4 – Space Invaders: Player Bullets

Exercises

- Recommended Reading:
  - Whitehead, T. (2014). *Donkey Kong Country's fate was determined by a risky Rare investment*. Retrieved from http://www.nintendolife.com/news/2014/02/donkey_kong_countrys_fate_was_determined_by_a_risky_rare_investment/

Exercises

- Recommended Reference Books:
  - Kelly, C. (2012) *Programming 2D Games*. Portland, OR: Ringgold Inc.
  - Harbour, J. (2009). *Advanced 2D Game Development*. Boston, MA: Cengage Learning.
  - Pavleas, J., Chang, J., Sung, K., & Zhu, R. (2013). *Learn 2D Game Development with C#*. New York, NY: Apress.
  - Ericson, C. (2005). *Real-Time Collision Detection*. San Francisco, CA: Morgan Kaufmann.

Exercises

- Recommended Reference Books:
  - Tremblay, C. (2004). *Mathematics for Game Developers*. Boston, MA: Thomson Course Technology PTR.
  - Dunn, F. & Parberry, I. (2002). *3D Mathematics for Graphics and Game Development*. Plano, TX: Wordware Publishing, Inc.
  - Schwarzl, T. (2012). *2D Game Collision Detection: An introduction to clashing geometry in games*. CreateSpace Independent Publishing Platform.

Summary

- Graphical Effects for Computer Games
- 2D Graphics
  - Frame Buffer, Sprites, Pixel Art
  - User Interface, Heads-up-display
- 2D Game Framework
  - Roll-our-own C++ 2D Graphics Framework
  - Sprites, Input, Entities, Collisions, Debugging…
- Exercises