UNIVERSITY OF Hull

Rather Useful Seminars

Three Thing Game and MonoGame

Rob Miles
Department of Computer Science

Getting Started on Windows 8.1

1. Install Visual Studio 2013
2. Install XNA 4.0

   https://msxna.codeplex.com/
3. Install MonoGame

   http://www.monogame.net/

- You can use this installation to create games for Windows Desktop and also ones that can be published in the Windows Store

Making a MonoGame Project

- Monogame adds new project types

MonoGame Sample Applications

- If you are new to development you might want to start from one of my sample applications
- These are guaranteed to work

MonoGame Sample Applications

- If you are new to development you might want to start from one of my sample applications
- These are guaranteed to work
  - Mostly

Cracker Chase

- Cracker Chase is not a very good game
- Chase the crackers with the cheese

How Games Work

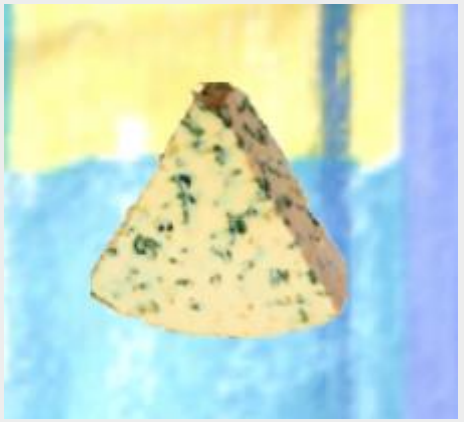
- Every game that has ever been written has these fundamental behaviours:
- Initialise all the resources at the start
  - fetch all textures, models, scripts etc
- Repeatedly run the game loop:
  - Update the game world
    - read the controllers, update the state and position of game elements
  - Draw the game world
    - render the game elements on the viewing device

Cracker Chase Behaviours

- The CrackerChase game does three things
- It loads the content
  - It updates the game
  - It draws the game
- We can take a look at how these work

A Word about Sprites

- A sprite is a game object that has a position and a texture
  - Plus a few other things
- We create these to represent game objects



Sprite Data

```
class Sprite
{
    protected int screenWidth;
    protected int screenHeight;

    protected Texture2D texture;
    protected Rectangle rectangle;

    protected float xPosition;
    protected float yPosition;

    protected float xResetPosition;
    protected float yResetPosition;
}
```

- This is all the data held in a standard sprite

Sprite Data

```
class Sprite
{
    protected int screenWidth;
    protected int screenHeight;

    protected Texture2D texture;
    protected Rectangle rectangle;

    protected float xPosition;
    protected float yPosition;

    protected float xResetPosition;
    protected float yResetPosition;
}
```

- The size of the screen it is part of

Sprite Data

```
class Sprite
{
    protected int screenWidth;
    protected int screenHeight;

    protected Texture2D texture;
    protected Rectangle rectangle;

    protected float xPosition;
    protected float yPosition;

    protected float xResetPosition;
    protected float yResetPosition;
}
```

- The texture that contains the sprite image

Sprite Data

```
class Sprite
{
    protected int screenWidth;
    protected int screenHeight;

    protected Texture2D texture;
    protected Rectangle rectangle;

    protected float xPosition;
    protected float yPosition;

    protected float xResetPosition;
    protected float yResetPosition;
}
```

- A rectangle that defines the width and height of the sprite

Sprite Data

```
class Sprite
{
    protected int screenWidth;
    protected int screenHeight;

    protected Texture2D texture;
    protected Rectangle rectangle;

    protected float xPosition;
    protected float yPosition;

    protected float xResetPosition;
    protected float yResetPosition;
}
```

- The X and Y position of the sprite on the screen

Sprite Data

```
class Sprite
{
    protected int screenWidth;
    protected int screenHeight;

    protected Texture2D texture;
    protected Rectangle rectangle;

    protected float xPosition;
    protected float yPosition;

    protected float xResetPosition;
    protected float yResetPosition;
}
```

- The place to put it back to when the game resets

Creating a background sprite

```
Texture2D cloth = Content.Load<Texture2D>("Tablecloth");
background = new Sprite(screenWidth,
                        screenHeight,cloth,screenWidth,0,0);
```

- This is how I make a simple sprite for the background
- I want the sprite to be the width of the screen, and placed at 0,0

Creating a background sprite

```
Texture2D cloth = Content.Load<Texture2D>("Tablecloth");
background = new Sprite(screenWidth,
                        screenHeight,cloth,screenWidth,0,0);
```

- The game sets these values so that the game will automatically fit any sized screen

Creating a background sprite

```
Texture2D cloth = Content.Load<Texture2D>("Tablecloth");
background = new Sprite(screenWidth,
                        screenHeight,cloth,screenWidth,0,0);
```

- The background sprite has a reset position of 0,0 so that it is displayed over the whole screen

Lists of Sprites

```
List<Sprite> gameSprites = new List<Sprite>();

…

gameSprites.Add(background);
```

- The game contains a list of sprites that are all the objects on the screen
- Each time I create a new sprite I add one to the list

Other kinds of sprites

```
cracker = new Target(screenWidth, screenHeight,
                     crackerTexture, crackerWidth, 0, 0);
gameSprites.Add(cracker);
```

- The cracker I am chasing is a Target sprite
- This is like a Sprite, but can position itself randomly on the screen

Size Calculation

```
int crackerWidth = screenWidth/20;
```

- The game calculates the width of the cracker as a fraction of the size of the screen
- This makes the game work on any sized display

Other kinds of sprites

```
cheese = new Mover(screenWidth, screenHeight, cheeseTexture,
        cheeseWidth, screenWidth / 2, screenHeight / 2, 500, 500);
```

- A Mover sprite can be made to move around the screen

Other kinds of sprites

```
cheese = new Mover(screenWidth, screenHeight, cheeseTexture,
        cheeseWidth, screenWidth / 2, screenHeight / 2, 500, 500);
```

- A Mover sprite can be made to move around the screen
- When we create a mover we tell it how fast it is allowed to move
  - The units are pixels per second

Game Behaviours

- A game has update and draw behaviours
- These map onto the methods in the XNA game class
- What happens in the methods depends on the state of the game
  - Starting the game
  - Playing the game

The Update method

```
protected override void Update(GameTime gameTime)
{
    switch (state)
    {
        case GameStates.Start_Screen:
            updateStartScreen(gameTime);
            break;
        case GameStates.Playing_Game:
            updateGamePlay(gameTime);
            break;
    }
}
```

- The Update method is called 60 times a second when the game is running

The Start Screen Update method

```
void updateStartScreen(GameTime gameTime)
{
    KeyboardState keys = Keyboard.GetState();

    if (keys.IsKeyDown(Keys.Space))
        startPlayingGame();
}
```

- The update method for the startscreen state is looking for the player to press the space key

The Start Screen Update method

```
void updateStartScreen(GameTime gameTime)
{
    KeyboardState keys = Keyboard.GetState();

    if (keys.IsKeyDown(Keys.Space))
        startPlayingGame();
}
```

- This is how a game will test the keyboard
- This code looks for a space key
- You can look for any key

Starting to Play the Game

```
void startPlayingGame()
{
    foreach (Sprite s in gameSprites)
        s.Reset();

    timer = 600;
    score = 0;

    state = GameStates.Playing_Game;
}
```

- When the game starts we reset each sprite and then set the score and timer values

Starting to Play the Game

```
void startPlayingGame()
{
    foreach (Sprite s in gameSprites)
        s.Reset();

    timer = 600;
    score = 0;

    state = GameStates.Playing_Game;
}
```

- This loop works through all the sprites and resets them all

Starting to Play the Game

```
void startPlayingGame()
{
    foreach (Sprite s in gameSprites)
        s.Reset();

    timer = 600;
    score = 0;

    state = GameStates.Playing_Game;
}
```

- The XNA clock ticks 60 times a second so this will give me 10 seconds of gameplay

Starting to Play the Game

```
void startPlayingGame()
{
    foreach (Sprite s in gameSprites)
        s.Reset();

    timer = 600;
    score = 0;

    state = GameStates.Playing_Game;
}
```

- Each time we eat a cracker we get 10 points

Starting to Play the Game

```
void startPlayingGame()
{
    foreach (Sprite s in gameSprites)
        s.Reset();

    timer = 600;
    score = 0;

    state = GameStates.Playing_Game;
}
```

- This starts the game playing by changing the state

Gameplay Update – cheese

```
void updateGamePlay(GameTime gameTime)
{
    KeyboardState keys = Keyboard.GetState();

    if (keys.IsKeyDown(Keys.Up))
        cheese.StartMovingUp();
    else
        cheese.StopMovingUp();

    ...
}
```

- This is part of the gameplay update method
- It tells the cheese to start moving if a key is pressed

Gameplay Update : sprites

```
void updateGamePlay(GameTime gameTime)
{
    ...
    foreach (Sprite s in gameSprites)
        s.Update(1.0f / 60.0f);
    ...
}
```

- This part of the update method updates each sprite

Gameplay Update : sprites

```
void updateGamePlay(GameTime gameTime)
{
    ...
    foreach (Sprite s in gameSprites)
        s.Update(1.0f / 60.0f);
    ...
}
```

- This tells the sprite that a 60th of a second has gone by since the last update
  - We change this if the speed of the game changes

Gameplay Update : Targets

```
void updateGamePlay(GameTime gameTime)
{
    ...
    foreach(Target t in crackers)
    {
        if(cheese.IntersectsWith(t))
        {
            BurpSound.Play();
            t.RandomPlace();
            score = score + 10;
        }
    }
}
```

- This code checks to see if the cheese has hit a cracker

Gameplay Update : Targets

```
void updateGamePlay(GameTime gameTime)
{
    ...
    foreach(Target t in crackers)
    {
        if(cheese.IntersectsWith(t))
        {
            BurpSound.Play();
            t.RandomPlace();
            score = score + 10;
        }
    }
}
```

- If it has we play a burp sound and update the score

Gameplay Update: state

```
void updateGamePlay(GameTime gameTime)
{
    ...
    timer = timer - 1;

    int secsLeft = timer/60;
    messageString = "Time: " + secsLeft.ToString() + " Score: " + score;

    if (timer == 0)
        gameOver();

}
```

- This updates the timer and status
- When the timer reaches 0 the game ends

Drawing the game world

```
void drawGamePlay()
{
    spriteBatch.Begin();

    foreach (Sprite s in gameSprites)
        s.Draw(spriteBatch);

    float xPos = (screenWidth - messageFont.MeasureString(messageString).X) / 2;

    Vector2 statusPos = new Vector2(xPos, 10);

    spriteBatch.DrawString(messageFont, messageString, statusPos, Color.Red);

    spriteBatch.End();
}
```

- There are separate draw methods for the gameplay and start modes

Drawing the game world

```
void drawGamePlay()
{
    spriteBatch.Begin();

    foreach (Sprite s in gameSprites)
        s.Draw(spriteBatch);

    float xPos = (screenWidth - messageFont.MeasureString(messageString).X) / 2;

    Vector2 statusPos = new Vector2(xPos, 10);

    spriteBatch.DrawString(messageFont, messageString, statusPos, Color.Red);

    spriteBatch.End();
}
```

- Each sprite is asked to draw itself

Sprite Drawing

```
public virtual void Draw(SpriteBatch spriteBatch)
{
    rectangle.X = (int)Math.Round(xPosition);
    rectangle.Y = (int)Math.Round(yPosition);
    spriteBatch.Draw(texture, rectangle,
                     Color.White);
}
```

- The sprite uses the data it holds to draw the texture at the right place on the screen

Making the Game

- The game is very simple, but can be expanded by adding more items
- We can also give items behaviours very easily
- The game has been structured so that it is easy to make new types of sprite
- Sample code at:

  www.ratherusefulseminars.com

Also available....

www.robmiles.com