Multiplatform Open Source Game Development with

MonoGame

Dominique Louis, Kenneth Pouncey, Dean Ellis

Dominique Louis (iOS)

- Doing Open Source for many years
- Maintained JEDI-SDL bindings(Delphi)
- DirectX bindings for Delphi
- Works on a couple of commercial games
  - Siege of Avalon (Now OpenSource - port to C#??)
  - Hero X
- Ran DelphiGamer.com with Dean way back when (last century)
- Played with XNA as back in 2007
- Currently working in digital signage (C#, Flex(yuk), Delphi)
- Hoping to complete my first marathon (London) in April :)
- **On shoulders of giants**

Kenneth Pouncey (MonoMac)

- Luxembourg based developer of the Open Source MonoGame project.
- My daytime job is as a systems analyst for financial institutions in various capacities from IBM Main Fame and IBM AS400 systems to PC applications written for Server and Desktop applications.
- Have worked on and contributed to multiple OpenSource projects ranging from Apache to KDE as well as having an OpenSource 5250 emulator project for IBM's AS400 called TN5250j written in Java.
- Became interested with MonoGame while contributing to MonoMac. Answering questions and implementing certain features in MonoMac for Dominique persuaded me to take a look at what sounded like an interesting project, at that time titled XNATouch. From there I picked up the Mac OSX port of MonoGame.

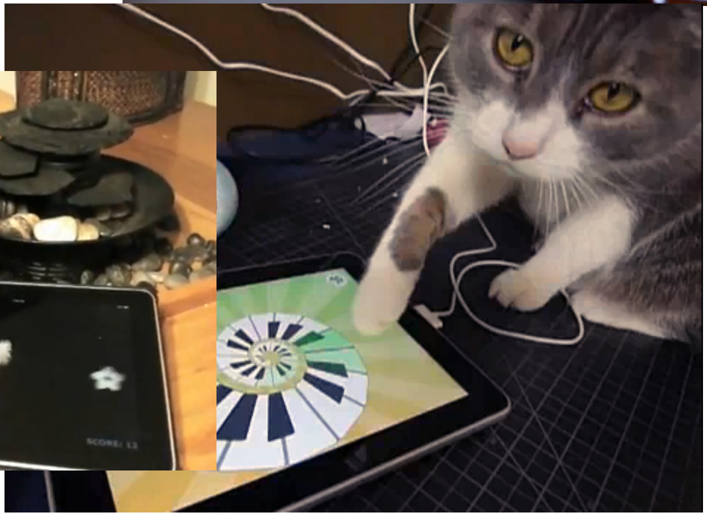
Dean Ellis (Android)

- Worked on OpenSource projects
  - JEDI-SDL
  - Delphi DirectX
  - Moonlight
- Works on Face Recognition Kiosk Devices during the day
  - Uses Moonlight in Desktop mode.
- Works primarily on the Android Port of MonoGame.

What is MonoGame?

- Open Source implementation of XNA 4.0 API
  - OpenGL, OpenTK, OpenAL
- Massively Multi-Platform Game Development
- Write Once, Play Everywhere.

What Platforms?



What is this XNA thing

- Microsoft Game API
  - Supports Windows/XBox360/Zune/Windows Phone 7
  - DirectX9
  - C#
- Free Development Tools
- Supports
  - 2D/3D
  - Networking
  - Sound
  - and much more.....

XNA is Serious Business

Bastion, 500k+ sold

Magicka, 1.1m+ sold

1m+ sold

History of MonoGame

- XNATouch on codeplex.com(José Antonio Leal de Farias (Jalfx))
    - XNA, MonoXNA, SilverSprite
- Wanted to get into the App gravy train
- v1.0 was iOS only, by 1.5 (Mac and Android)
- Just after v1.5, **Geoff Norton (Novell)** @geoffnorton helped us migrate to github
- v2.0 release and backing from Team Xamarin helped it get massive visibility.

Cool...What can MonoGame do?

- 2D Graphics
- Custom Effects (GLSL)
- XNA Content Files
- Networking (lidGren)
  - Local Only
- Sound/Music(via OpenAL - not Android)
- XACT (proprietary audio by Microsoft)

Cool...What can MonoGame do?

- Video Playback
  - as per platform
- Native asset loading
  - png/jpeg/gif/tiff/pdf
  - wav/mpeg
- Input
  - Gamepad on Window/Linux/MacOS
  - Gestures on iOS/Android

Harder than it should be

- Have to support many undocumented file formats for binary compatibility
  - Content serialisation+compression
  - DirectX Effect objects
  - XACT project files
  - xWMA+XMA audio codecs

What are we missing?

- Custom Content Pipeline
  - Cannot generate content files
- 3D Support (very soon)
- HLSL Effects (very soon)
- Networking (?)
  - Limited to Local Networks
  - Cannot link with XNA based games
- Many bits and pieces

How you can help?

- MonoDevelop
  - MacOS
  - Linux
  - Android (MacOS, Windows)
  - iOS (Mac OS X and a **Mac**)
- Visual Studio 2010
  - Android
  - Windows Phone 7
- Download MonoGame from github.com
  - https://github.com/mono/MonoGame/zipball/2.1.0.0
- Download Samples from github.com
  - https://github.com/CartBlanche/MonoGame-Samples/downloads

How we work?

- 8 core maintainers
- All pull requests are peer reviewed
- If you submit enough patches...
- Automated Tests
  - Run against XNA and MonoGame to ensure conformance
  - Run unattended
- On Demand Integration Build and Testing system (via IRC).
  - Queries the GitHub API
  - Runs on Linux, Mac and Windows
  - iOS and Android are in development

Lots of Sample Code

- Samples Repository contains lots of examples
  - XNA AppHub Samples
  - Windows Phone 7
  - MonoGame Team
- Exampes available for each Platform
  - 38 samples for MacOSX
  - 25 samples for iOS
  - 14 samples for Linux
  - 11 samples for Android
- New Samples are added all the time
- Starter Packs from AppHub

Demos

- Vector Rumble
- Catapult (mobile)
- VideoPlayer
- Role Playing Game

Adding Ads to you Apps - Android

- Original Code by Greg Shackles - http://www.gregshackles.com/2011/02/using-admob-in-a-monodroid-application/
- Adding Ads to apps can be tricky
- The Basic idea is
    - Add the Admob sdk to your project
    - Add the AdmobHelper.java
    - Add the AdMobHelper.cs
    - Set the TargetFramework to 4.0
    - Add the required xml to the AndroidManifest.xml
    - Replace the normal MonoGame startup code with one that uses a FrameLayout.
- Source will be available on GitHub in the Samples Repo

AndroidManifest.xml Changes

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        android:versionCode="1"
        android:versionName="1.0"
        package="MyGame.Android">
  <application android:label="MyGame" android:icon="@drawable/icon">
    <activity
      android:name="com.google.ads.AdActivity"
      android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreenSize"/>
  </application>
  <uses-sdk android:minSdkVersion="8" />
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
</manifest>
```

Activity Code Changes

```
View adView;
protected override void OnCreate(Bundle bundle)
{
MyGame.Activity = this;
var g = new MyGame();
FrameLayout fl = new FrameLayout(this);
fl.AddView(g.Window);
adView = AdMobHelper.CreateAdView(this, "publisherid");
// AdMobHelper.AddTestDevice(adView,"deviceid");
fl.AddView(adView);
SetContentView(fl);
g.Run();
}
```

Things to watch out for

- Cannot Debug Application
  - You need to have a device running 4.0> to debug the app.
  - You can run release build packages on older devices
- Sometimes ads do not appear
  - This usually because there are no ads available.
  - Only occurs on new accounts
- Cannot position ads
  - Currently android.view.GRAVITY is not available
  - look up the values on android developer documents

Adding Ads to your Apps - iOS

https://iad.apple.com/itcportal/#app_homepage

```
using MonoTouch.iAd;

if (UIDevice.CurrentDevice.UserInterfaceIdiom ==
   UIUserInterfaceIdiom.Phone)
{
   bannerHeight = 50;
}
else if (UIDevice.CurrentDevice.UserInterfaceIdiom ==
        UIUserInterfaceIdiom.Pad)
{
   bannerHeight = 66;
}
```

Adding Ads to your Apps - iOS

```
var adView = new ADBannerView();
NSMutableSet nsM = new NSMutableSet()
// Multi-orientation
// nsM.Add(ADBannerView.SizeIdentifierLandscape);
nsM.Add(ADBannerView.SizeIdentifierPortrait);
adView.RequiredContentSizeIdentifiers = nsM;

adView.AdLoaded += delegate(object sender, EventArgs e)
{
   adView.Hidden = false;
   adView.Frame = new RectangleF(
      0, (int)UIScreen.MainScreen.Bounds.Height - bannerHeight,
      dvcMain.View.Frame.Width, adView.Frame.Height);
};

adView.FailedToReceiveAd += delegate(object sender, AdErrorEventArgs e)
{
   Console.WriteLine(e.Error);
   adView.Hidden = true;
};

iOSGameWindow.View.AddSubview(adView);
```

Network Games in MonoGame

- Goals
  - Implement as close as possible the networking functionality of XNA
  - Make it as painless as possible for developers to port already working networked projects to MonoGame

Getting Started With Networked Games

- Initalize the gamer services subsystem
- Most games will use the **GamerServicesComponent**
- Update at regular intervals.
- To start using the Gamer Services all one must do is to add one line to the Game constructor:

```
// Initialize GamerServiceComponent by adding it to our
// game's components
Components.Add(new GamerServicesComponent(this));
```

Network Session Management

- NetworkSession.Update
    - Sends network packets.
    - Changes session state such as which players are in the session.
    - Raises managed events for any significant state changes.
    - Returns incoming packet data.
- This allows developers to program against the network session object without any threading concerns.
- Session updates are kept separate from the gamer services system pumping

```
// Pump the underlying session object.
NetworkSession.Update();
```

Matchmaking

- Publish an instance of the game for others to discover and join the game. Available through NetworkSession.Create and NetworkSession.BeginCreate
- Discovering an instance of a game to join. Available through the NetworkSession.Find and NetworkSession.BeginFind methods, which return a collection of AvailableNetworkSession instances.
- AvailableNetworkSession exposed properties.

```
public static AvailableNetworkSessionCollection Find(
   NetworkSessionType sessionType,
   int maxLocalGamers,
   NetworkSessionProperties searchProperties);

AvailableNetworkSessionCollection availableSessions;
int maximumLocalPlayers = 1;
availableSessions = NetworkSession.Find(
   NetworkSessionType.SystemLink,
   maximumLocalPlayers,
   searchProperties);
```

Players

- Exposed from the NetworkSession object, along with events indicating when players join or leave the session.
- Determine which player is host (NetworkSession.Host) and which players are local (NetworkGamer.IsLocal).

```
// Listen for invite notification events.
NetworkSession.InviteAccepted += (sender, e) =>
   NetworkSessionComponent.InviteAccepted(screenManager, e);

if (NetworkSession.IsHost) {
   // Program logic here only if we are the hosting session
}
```

Players Joining and Leaving

- A NetworkSession contains several properties which list the players who have joined a session.
  - NetworkSession.AllGamers
  - NetworkSession.LocalGamers
  - NetworkSession.RemoteGamers
- To respond to players joining or leaving the game
  - NetworkSession.GamerJoined
  - NetworkSession.GamerLeft

```
// set the networking events
networkSession.GamerJoined += gamerJoinedHandler;
networkSession.GameStarted += gameStartedHandler;
networkSession.SessionEnded += sessionEndedHandler;
```

Create a Network Session

- There are multiple types of sessions that can be created
    - Local sessions
    - Local network sessions
    - Live sessions, over the internet, are still not implemented in MonoGame.
- Creating session is called the host, and owns the multiplayer session
    - Host does not imply a game server or authority.
    - No implied network topology
    - Network topology determined by your game implementation

```
NetworkSession session;
int maximumGamers = 8; // The maximum supported is 31
int privateGamerSlots = 2;
int maximumLocalPlayers = 1;

// Create the session
session = NetworkSession.Create(
    NetworkSessionType.SystemLink,
    maximumLocalPlayers, maximumGamers,
    privateGamerSlots, sessionProperties);
```

To Subscribe to Session Events

- Subscribe to any session events that your game is interested in
- Multiplayer session events
  - GameStarted and GameEnded
  - GamerJoined and GamerLeft
  - SessionEnded

```
session.GamerJoined += new EventHandler<GamerJoinedEventArgs>(session_GamerJoined);
session.GamerLeft += new EventHandler<GamerLeftEventArgs>(session_GamerLeft);
```

Programmer defined function

Send data to all peers

- Create a PacketWriter to use in writing the data.
- Loop through the LocalGamers collection and send data to all peers.
- Call the various overloads of the PacketWriter.Write method to store data into the writer
- Pass the PacketWriter to SendData
- Sending the packet will automatically clear the PacketWriter

Note: SendDataOptions passed to SendData
- ○ Choose the value for options that are appropriate for the type of data being sent.
- ○ Not all game data needs to be sent reliably,
- ○ Sending excessive data using SendDataOptions. ReliableInOrder can cause client to lag

Send data to all peers

```
// Our game defined Message Types to be sent
enum MessageType : byte
{
    NewGame = 1,           // New game was started
    CatapultFiring = 2,    // One of our Catapults is Firing
    CatapultAiming = 3,    // One of our Catapults is taking Aim
    UpdateEnvironment = 4, // Update our environment. Clouds, Wind speed...
}
```

```
if (NetworkSession.IsHost) {
    packetWriter.Write((int)MessageType.UpdateEnvironment);
    packetWriter.Write(wind);
    packetWriter.Write(cloud1Position);
    packetWriter.Write(cloud2Position);

    // Update our environment variables, and send their
    // latest position data to everyone in the session.
    foreach (LocalNetworkGamer gamer in NetworkSession.LocalGamers) {
        gamer.SendData(packetWriter, SendDataOptions.ReliableInOrder);
    }
}
```

How to receive data

- Create a PacketReader to assist in reading the incoming network data
- To read a packet, pass the PacketReader to ReceiveData
- Use the various PacketReader.Read methods to extract data from the reader.

How to receive data

```
void ReadIncomingPackets(LocalNetworkGamer gamer)
{
   // Keep reading as long as incoming packets are available.
   while (gamer.IsDataAvailable) {
      // Read a single packet from the network.
      NetworkGamer sender;
      gamer.ReceiveData(packetReader, out sender);

  // Discard packets sent by local gamers:
      // we already know their state!
      if (sender.IsLocal)
         continue;

      MessageType msgType = (MessageType)packetReader.ReadInt32();
      switch (msgType) {
  case MessageType.NewGame:
  // ReceiveNewNetworkedGame();
  break;
      case MessageType.CatapultAiming:
         if (isFirstPlayerTurn && !NetworkSession.IsHost) {
  playerOne.Catapult.CurrentState = CatapultState.Aiming;
            playerOne.isDragging = true;

  catapultInfoVector = packetReader.ReadVector3();

  playerOne.Catapult.ShotStrength = catapultInfoVector.X;
  playerOne.Catapult.ShotVelocity = catapultInfoVector.Y;
  playerOne.ArrowScale = catapultInfoVector.Z;
         }

         if (!isFirstPlayerTurn && NetworkSession.IsHost) {
            playerTwo.Catapult.CurrentState = CatapultState.Aiming;
            playerTwo.isDragging = true;

            catapultInfoVector = packetReader.ReadVector3();

  playerTwo.Catapult.ShotStrength = catapultInfoVector.X;
            playerTwo.Catapult.ShotVelocity = catapultInfoVector.Y;
  playerTwo.ArrowScale = catapultInfoVector.Z;
         }
         break;
```

```
      case MessageType.CatapultFiring:

         if (isFirstPlayerTurn  && !NetworkSession.IsHost) {
            catapultInfoVector = packetReader.ReadVector3();
            playerOne.Catapult.Fire (catapultInfoVector.Y);
            playerOne.Catapult.CurrentState = CatapultState.Firing;
            playerOne.ResetDragState();
         }
         if (!isFirstPlayerTurn  && NetworkSession.IsHost) {
            catapultInfoVector = packetReader.ReadVector3();
            playerTwo.Catapult.Fire (catapultInfoVector.Y);
            playerTwo.Catapult.CurrentState = CatapultState.Firing;
            playerTwo.ResetDragState();
         }
         break;
      case MessageType.UpdateEnvironment:
         wind = packetReader.ReadVector2();
         cloud1Position = packetReader.ReadVector2();
         cloud2Position = packetReader.ReadVector2();
         // Set new wind value to the players and
         playerOne.Catapult.Wind =
            playerTwo.Catapult.Wind =
               wind.X > 0 ? wind.Y : -wind.Y;
         break;
      }
   }
}
```

To End the Session

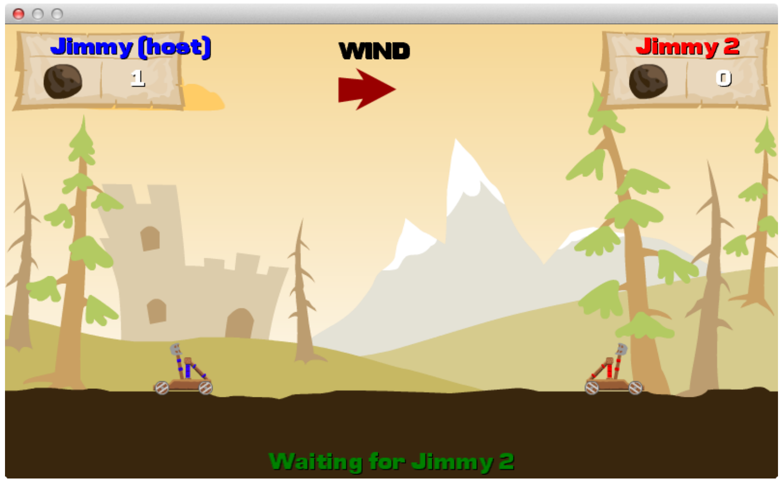
- NetworkSession.EndGame.
- The host checks if there are players remaining in the game before ending the session.

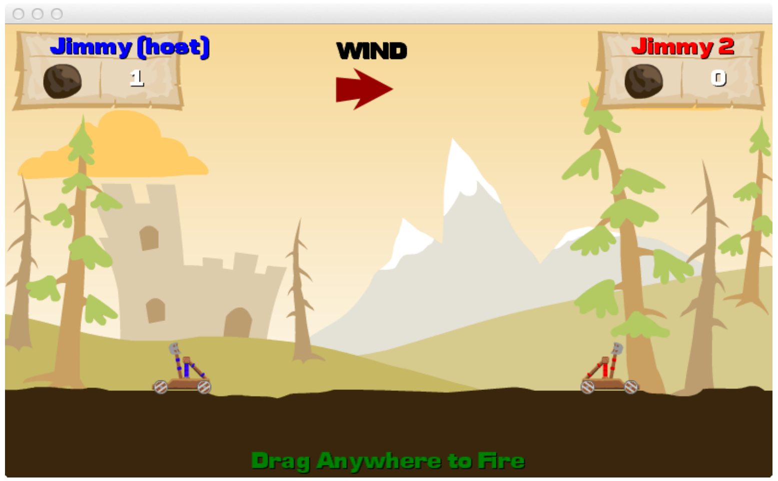
```
/// <summary>
/// Handles "Exit" menu item selection
/// </summary>
protected override void OnCancel(PlayerIndex playerIndex)
{
    // Tear down our network session
    var session = ScreenManager.Game.Services.GetService(typeof(NetworkSession)) as NetworkSession;
    if (session != null) {
        if (session.AllGamers.Count == 1) {
            session.EndGame();
        }
        session.Dispose();
        ScreenManager.Game.Services.RemoveService(typeof(NetworkSession));
    }
    AudioManager.StopSounds();
    ScreenManager.AddScreen(new MainMenuScreen(), null);
    ExitScreen();
}
```

Example Game

CatapultWars

CatapultNetWars





On the App Store

Many titles already using MonoGame:

iOS
- 28 titles!

Android
- 5 titles.

Mac OS, Linux
- Wizorb
- Unofficially, Terraria

Chrome
- Bastion

The Future

- MonoGame v2.5 very soon
  - Tonnes of bug fixes
  - Fully ES 2.0 (GLSL shaders!)

- MonoGame v3.0 soon, too
  - 3D API (Thanks to Inflight Dev Studio)
  - HLSL Shaders

MonoGame's Future

API

- Extended networking support
- DirectX 11 backend for Metro
- CellSDK(http://www.cellsdk.com/)
- Networking (alternative to Xbox Live)
- Built-in advertising support.

Platforms

- PlayStation Suite (started, but SDK is influx)
- Google 'Native Client' (Bastion)
- Raspberry Pi

Contact

Dominique Louis
  Twitter : @SoftSavage
  Email : savagesoftware@gmail.com

Kenneth Pouncey
  Twitter : @cocoamono
  Email : kjpou1609@gmail.com
Dean Ellis
  Twitter : @infspacestudios
  Email : dellis1972@googlemail.com

IRC
#monogame, irc.gnome.org

Questions ??

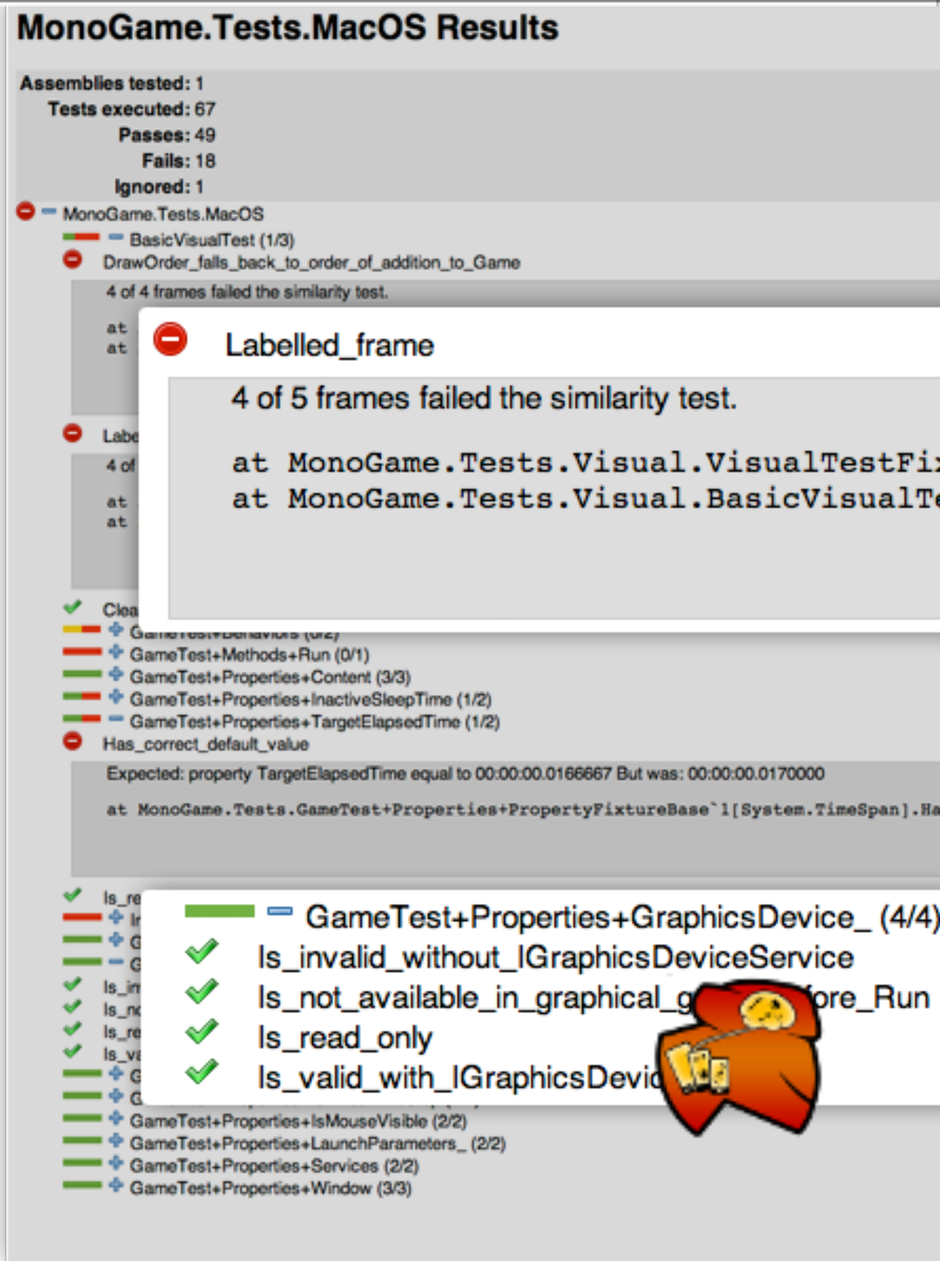
monogame.codeplex.com(discussions)

github.com/mono/MonoGame

Toward Stabler Development

Automated Tests

- Written with NUnit 2.5
- Run against XNA and MonoGame to ensure conformance
- Run unattended
- Ordinary unit tests to verify the functionality of properties and methods
- Visual tests
  - Verify rendering on all platforms to within a certain tolerance
  - Frames-of-interest are captured and compared to reference frames
    - Currently uses pixel-value diffs
    - Extensible with new frame comparers
    - Can aggregate the results of multiple frame comparers for a final result
    - More sophisticated comparisons are coming (e.g., edge-detection)



MonoGame.Tests.MacOS Results

```
Assemblies tested: 1
   Tests executed: 67
           Passes: 49
            Fails: 18
          Ignored: 1
MonoGame.Tests.MacOS
    BasicVisualTest (1/3)
  DrawOrder_falls_back_to_order_of_addition_to_Game
    4 of 4 frames failed the similarity test.

    at
    at

  Labe
    4 of

    at
    at

  Clea
    GameTest+Behaviors (0/2)
    GameTest+Methods+Run (0/1)
    GameTest+Properties+Content (3/3)
    GameTest+Properties+InactiveSleepTime (1/2)
    GameTest+Properties+TargetElapsedTime (1/2)
  Has_correct_default_value
    Expected: property TargetElapsedTime equal to 00:00:00.0166667 But was: 00:00:00.0170000

    at MonoGame.Tests.GameTest+Properties+PropertyFixtureBase`1[System.TimeSpan].Ha

  Is_re
    I
    G
    G
  Is_in
  Is_no
  Is_re
  Is_va
    G
    G
    GameTest+Properties+IsMouseVisible (2/2)
    GameTest+Properties+LaunchParameters_ (2/2)
    GameTest+Properties+Services (2/2)
    GameTest+Properties+Window (3/3)
```

```
Labelled_frame
  4 of 5 frames failed the similarity test.

  at MonoGame.Tests.Visual.VisualTestFi
  at MonoGame.Tests.Visual.BasicVisualTe
```

```
GameTest+Properties+GraphicsDevice_ (4/4)
  ✓ Is_invalid_without_IGraphicsDeviceService
  ✓ Is_not_available_in_graphical_g          ore_Run
  ✓ Is_read_only
  ✓ Is_valid_with_IGraphicsDevic
```

Low-Friction Testing

MonoGame Build Bot

- Queries the GitHub API
- Submits to the MonoGame Build Coordinator
- IRC
  - Hubot from the GitHub Team
  - For example

```
mgbot test develop
mgbot test @kjpou1 a840f0e...
mgbot test @mono pull 275
```

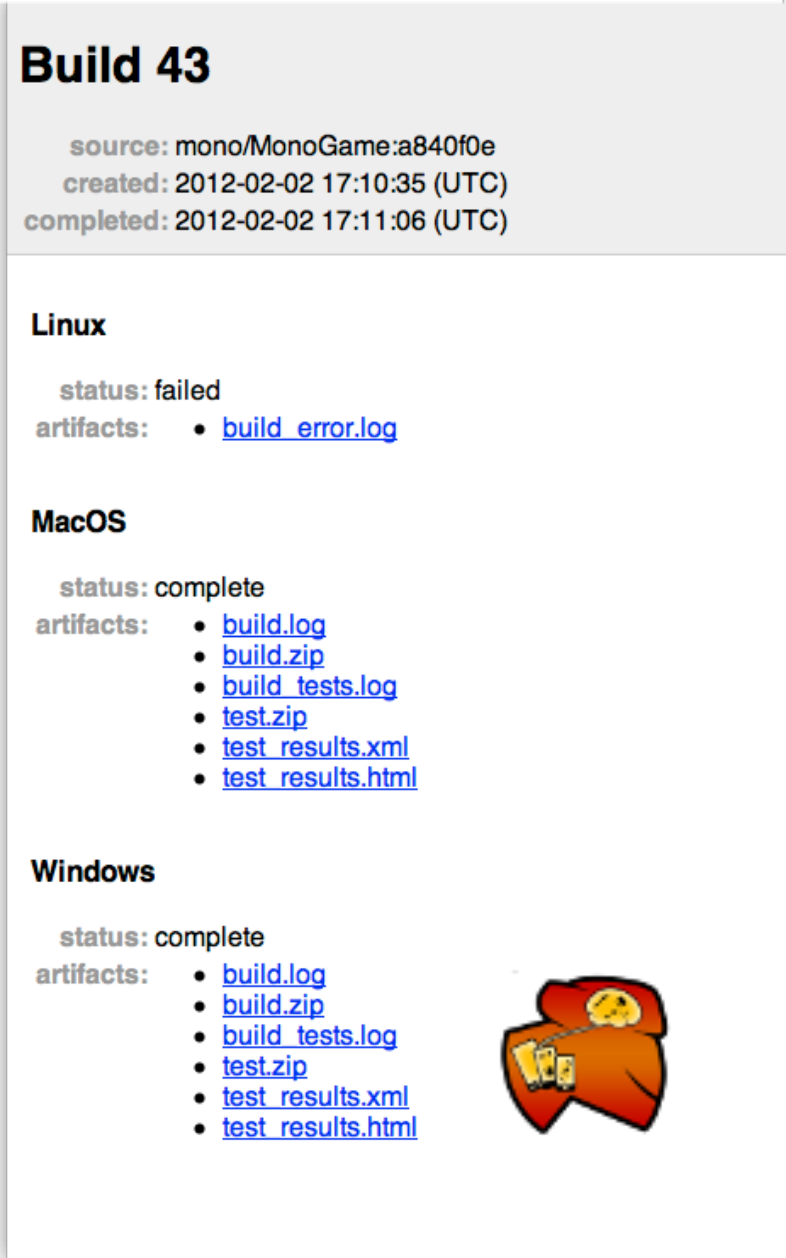
Distributed Builds and Tests

Build Coordinator

- Accepts new build requests (JSON)
- Distributes build requests to Build Nodes
- Collects and displays results from Build Nodes

Build Node

- Per-platform
  - Runs on Linux, Mac and Windows
  - iOS and Android are in development
- Polls the Build Coordinator for new builds
- Claims a build
- Fetches the source, applies patches
- Builds MonoGame
- Builds and runs the test assembly
- Submits results to the Build Coordinator



Build 43

source: mono/MonoGame:a840f0e
created: 2012-02-02 17:10:35 (UTC)
completed: 2012-02-02 17:11:06 (UTC)

Linux

status: failed
artifacts:
- build_error.log

MacOS

status: complete
artifacts:
- build.log
- build.zip
- build_tests.log
- test.zip
- test_results.xml
- test_results.html

Windows

status: complete
artifacts:
- build.log
- build.zip
- build_tests.log
- test.zip
- test_results.xml
- test_results.html