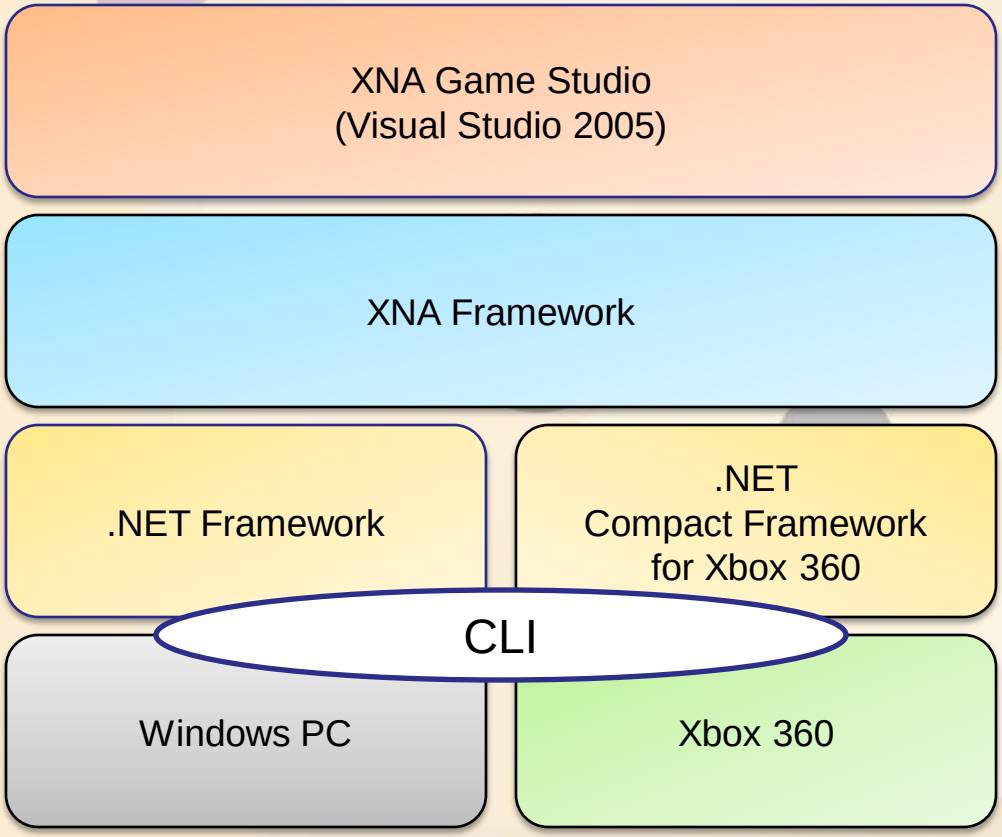
XNA Framework 2.0

M@STER SESSION 01

XNA Framework の技術的背景

- システム基盤は .NET Framework
- マネージ環境で実行
- クロスプラットフォーム
  - Windows, Xbox360, Zune
- DirectX, MDX から独立している

開発・実行環境の構造



XNA Game Studio

Visual Studio 2005 を拡張したゲーム開発環境。

コードの他に、画像などのコンテンツ管理機能も備わっている。

XNA Framework

クロスプラットフォームのゲーム開発用フレームワーク

CLI 共通言語基盤

ISO で承認されている国際標準規格

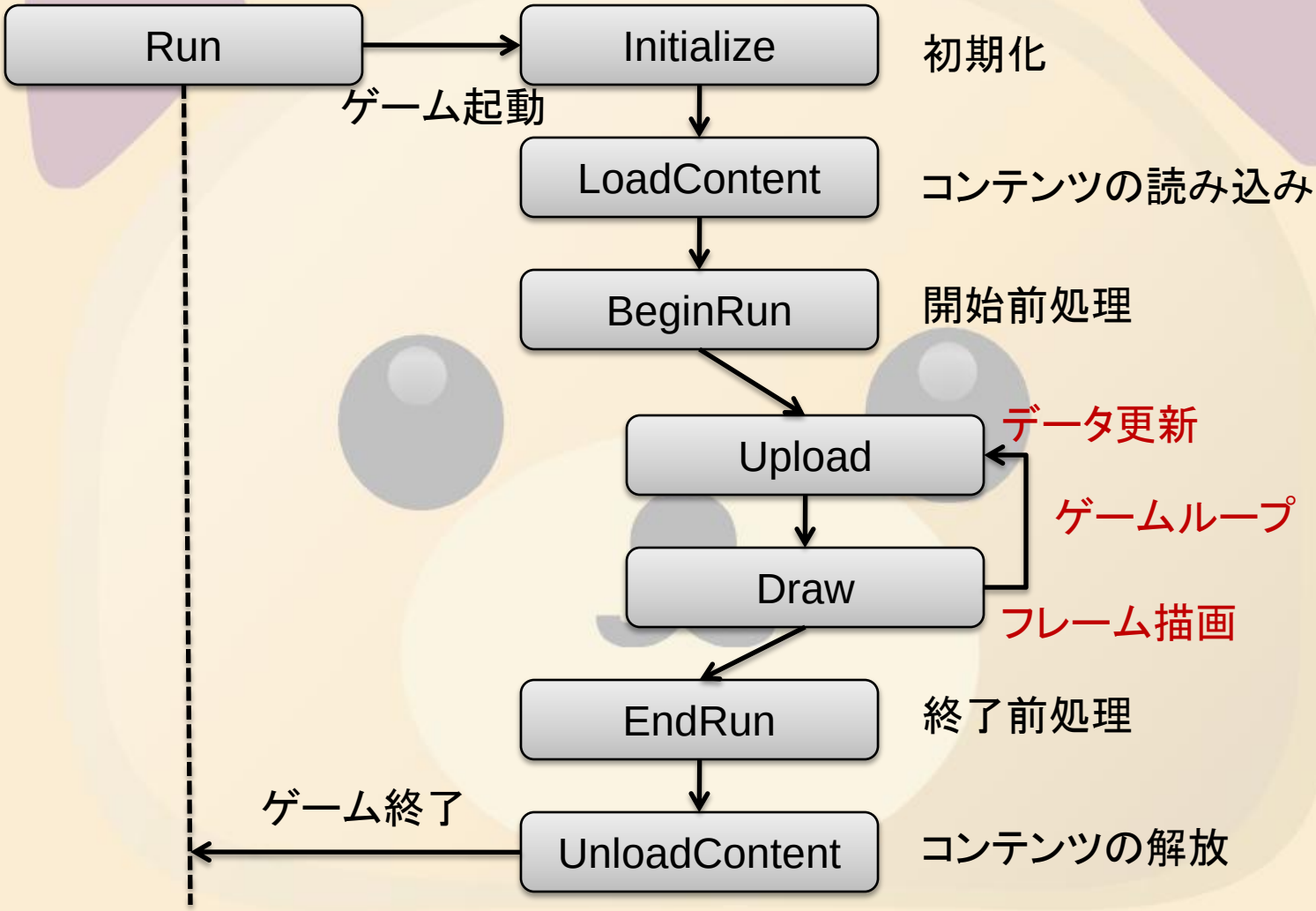
主な名前空間

- Microsoft.Xna.Framework
  - 基礎的なゲームの機能
- Microsoft.Xna.Framework.Graphics
  - グラフィック関連
- Microsoft.Xna.Framework.Content
  - コンテンツ管理
- Microsoft.Xna.Framework.Audio
  - オーディオ再生
- Microsoft.Xna.Framework.Input
  - コントローラ、マウス、キーボード入力
- Microsoft.Xna.Framework.Storage
  - ストレージ選択
- Microsoft.Xna.Framework.GamerServices
  - ゲーマーサービス
- Microsoft.Xna.Framework.Net
  - ネットワーク

ゲームの起動

- Game クラスを継承する
- コンストラクタ
  - 継承クラス固有の初期化
- Initialize() メソッド
  - Game に関連した初期化処理
- LoadContent() メソッド
  - リソース・コンテンツの生成・読み込み・初期化
- ゲームループ
  - Update() メソッド
    - ゲームデータの更新
  - Draw() メソッド
    - ゲームの描画
- UnloadContent() メソッド
  - リソース・コンテンツの破棄・解放

ゲームの流れ


Run
Initialize
初期化
ゲーム起動
LoadContent
コンテンツの読み込み
BeginRun
開始前処理
Upload
データ更新
ゲームループ
Draw
フレーム描画
EndRun
終了前処理
ゲーム終了
UnloadContent
コンテンツの解放

ゲームループ

- Update() メソッド
  - ゲームデータを「次」の状態に更新する
  - コントローラなどの入力を受ける
  - 関連するコンポーネント、機能の更新
  - 他のあらゆる処理よりも優先される
- Draw() メソッド
  - ゲーム画面（フレーム）を構築する
  - 処理が遅れている場合は省略される

固定と可変ステップ

- 固定ステップ
  - デフォルトの設定
  - 1フレームを呼び出す時間間隔を固定する
  - 常に一定の速度でゲームを進行できる
  - 処理が遅れた場合は Draw() を省略
- 可変ステップ
  - 可能な限り早い速度でフレームを描画する

パフォーマンス

- 安定したゲームループの実行が重要
- ゲームはリアルタイムシステム
  - 目標の時間内に処理を終わらせる
  - 秒 60 回の Draw() が目標
  - 1 フレームの描画は 16.6 ミリ秒
- メモリ管理が重要
  - インスタンス化と破棄の効率化
  - 変換処理の効率化

グラフィックス

- Microsoft.Xna.Framework.Graphics 名前空間
- 高度な3次元グラフィックス処理が可能
  - 基本的な機能は BasicEffect でサポート
  - HLSLを使用したエフェクトの記述
  - モデルの描画（X、FBX に対応）
- 擬似的に2次元グラフィックスを描画できる（スプライト）

画像の描画

- SpriteBatch クラス
  - 擬似的に、画像やフォントを描画する
  - Draw() メソッド
    - 画像を指定座標に描画する
  - DrawString() メソッド
    - 文字列を指定されたフォントで描画する
- Texture2D
  - ビットマップ画像を表す

デバイス管理

- GraphicsDevice の管理は複雑
  - 適切なデバイスの選択
  - デバイスの初期化・設定
  - 画面モードの遷移
- GraphicsDeviceManager
  - デバイスの生成・管理を代行
- 手動管理も可能だが、まずやらない

コンテンツ・パイプライン

- 事前コンパイルによる最適化
  - 画像・3Dモデル・音声・フォントなどには、ゲームに関係のないデータが含まれているため冗長である
  - ビルド時に、データをオブジェクト表現可能なバイナリに直列化し、実行時の負担を軽減する
  - Windows 用プロジェクトと Xbox 360 用プロジェクトでリソースを共有できる
  - 独自のデータ形式を組み込むことも可能
- アセット
  - ビルド時にリソースをコンパイルし、直列化したデータ
  - アセット名から、実行時にデータを読み込むことができる

アセットの読み込み

- ContentManager
  - ゲームのコンテンツを管理する
  - Load<T>() メソッド
    - 実行時にアセット名からファイルを読み込み、適切なオブジェクトとして返す

コンポーネント化とサービス

- 再利用可能なゲームの部品化
- ゲームコンポーネント
  - GameComponent クラスを継承して作成
  - データ更新、描画の移譲
  - ゲームに対し、任意に追加・削除できる
- ゲームサービス
  - サービスプロバイダの提供
  - 疎結合でコンポーネントを連携させる仲介機能

ゲーマーサービスの機能

- ガイドの提供
  - アカウントのサインイン・サインアウト
  - メッセージの表示
  - キーボード入力
  - ストレージの選択

ゲーマーサービスへの対応

- GamerServicesComponent
  - XNA Framework の開発モデルに従う
  - コンポーネントとしてゲームに登録可能
  - GamerServicesDispatcher のラッパー
- GamerServicesDispatcher
  - ゲーマーサービスの提供
  - Initialize() と Update() の呼び出しが必須
  - 開発モデルに依存しない

ネットワーク

- LAN (システムリンク)、Live に対応
- クロスプラットフォーム
- セッション単位による通信管理
  - ホストによるセッションの作成
  - セッションの検索・参加
- 信頼性のあるUDPプロトコル
  - パケット単位のデータ送受信
  - 低水準なパケット管理は不要
  - 信頼性の制御が可能

セッション

- プレイヤーを接続させる部屋
- ホスト
  - セッションを作成したゲーム
  - ゲームの状態、参加者数などを管理
  - 他の参加者の参加を待機
- セッションへの参加
  - セッションの検索
  - セッションの参加

セッション検索

- セッションは GUID に関連付けられる
  - 実行中のアセンブリと同一の GUID のゲームによって作られたセッションを検索する
  - GUID を一致させれば、異なるプロジェクトによって作られたゲームの間での通信も可能
- Xbox Live アーケードの制限
  - ゲームは GUID によって識別される
  - 同じ GUID のゲームを複数配置できない
  - 同一タイトルでも GUID が異なれば配置可能

データの送受信

- バイト配列の送受信
  - 最もシンプルな方法だが、データを直列化する必要がある。
- PacketWriter によるデータ送信
  - BinaryWriter を継承するクラス
  - 基本的なデータ型の Write() メソッドを実装
- PacketReader によるデータ受信
  - BinaryReader を継承するクラス
  - 基本的なデータ型の Read～() メソッドを実装

参考

- XNA デベロッパーセンター
  - http://www.microsoft.com/japan/msdn/xna/
- XNA Game Studio で作るマインスイーパ
  - http://www.microsoft.com/japan/msdn/vstudio/express/learn/xna/
- XNA Japan Team Blog
  - http://www.microsoft.com/japan/msdn/vstudio/express/learn/xna/
- XNA Team Blog (英語)
  - http://blogs.msdn.com/xna/