ゲーム開発モデルの基礎

独特なゲーム開発モデル

- ゲーム開発モデルは、一般的なアプリケーション開発モデルとは異なる
  - 同じソフトウェアでも、使われ方や求められる内容が大きく異なる
  - エンターテイメント性が強く「正常に動作するプログラム」というだけでは価値とされない
  - プランナー、プロデューサー、デザイナーなど、多くの非開発者が関係する

ゲーム開発から学ぶこと

- ユーザーエクスペリエンス
  - ゲーム業界にとっては「何をいまさら」というくらいユーザーエクスペリエンスの概念は常識
  - 業務分野でもXAMLによるUIデザインなど、コードからリソースを分離して分業開発しようという流れになっている
  - WPFなどUXを重視する次世代プラットフォームの深部は、ゲームのデザインに似てきている

業務アプリケーションの開発モデル

- イベント駆動
- 3層モデル
  - データソース
  - ビジネスロジック
  - プレゼンテーション
- 高度な仮想化・抽象化
- 保守性・ライフサイクルの重視

ゲームの開発モデル

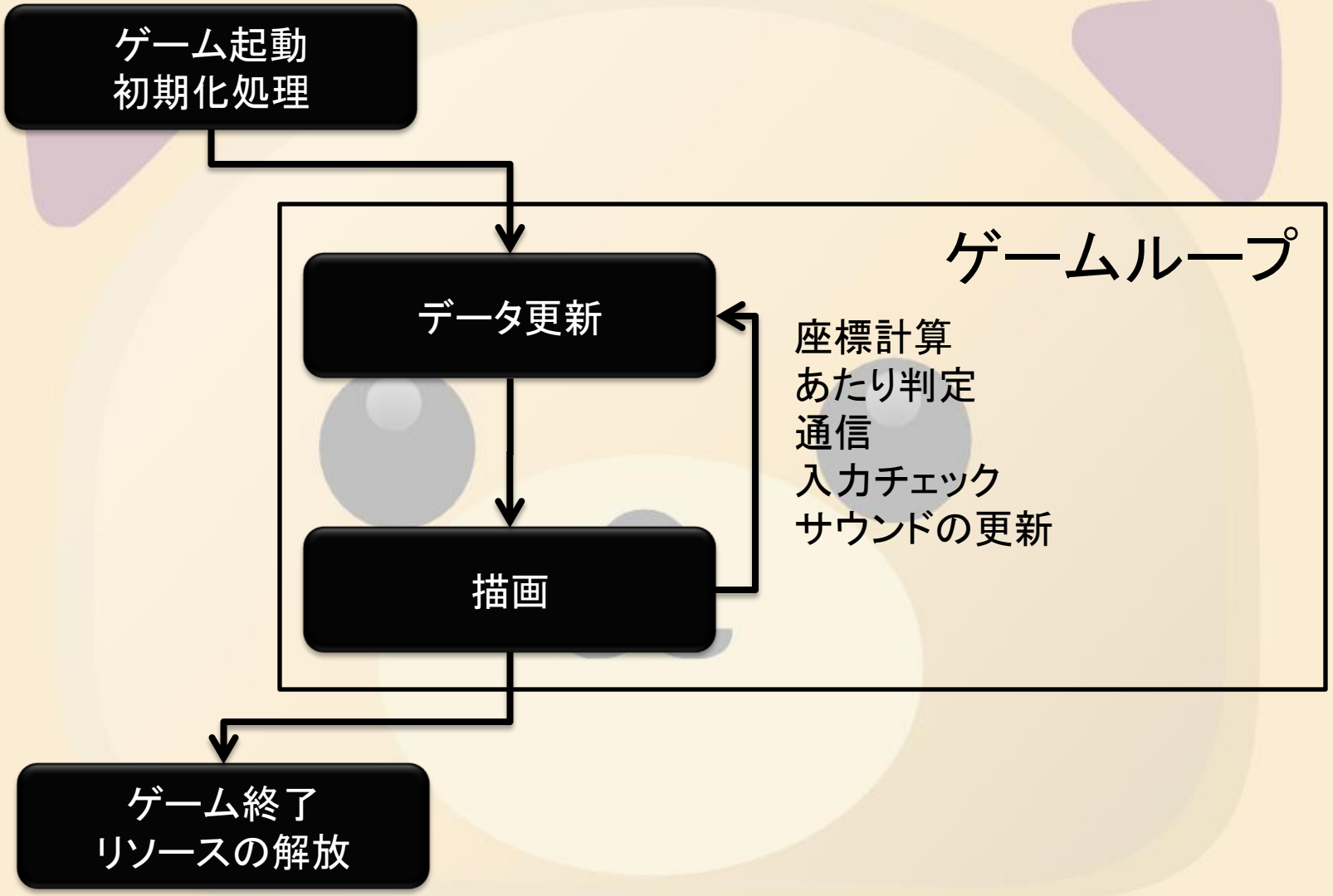
- ゲームループ（フレーム駆動）
- プレゼンテーションに特化した1層モデル
  - 実行時の不要な変換は避ける
  - コンパイル時にデータはすべて変換される
  - 読み込むデータはメモリイメージそのもの
- パフォーマンス重視
- 部品化と分業化

ゲーム開発の課題

- 柔軟で複雑なプレゼンテーション
- ハードウェアに依存しないゲーム進行
  - ミサイルのスピードが変化してはいけない
    - タイマを使うと精度によって速度が変わってしまう
- 部品化における責任の移譲
  - どのコードが･どこに描画するか
  - ゲームデータの統合的な管理の必要性
  - ゲームデータの水平分散的な利用

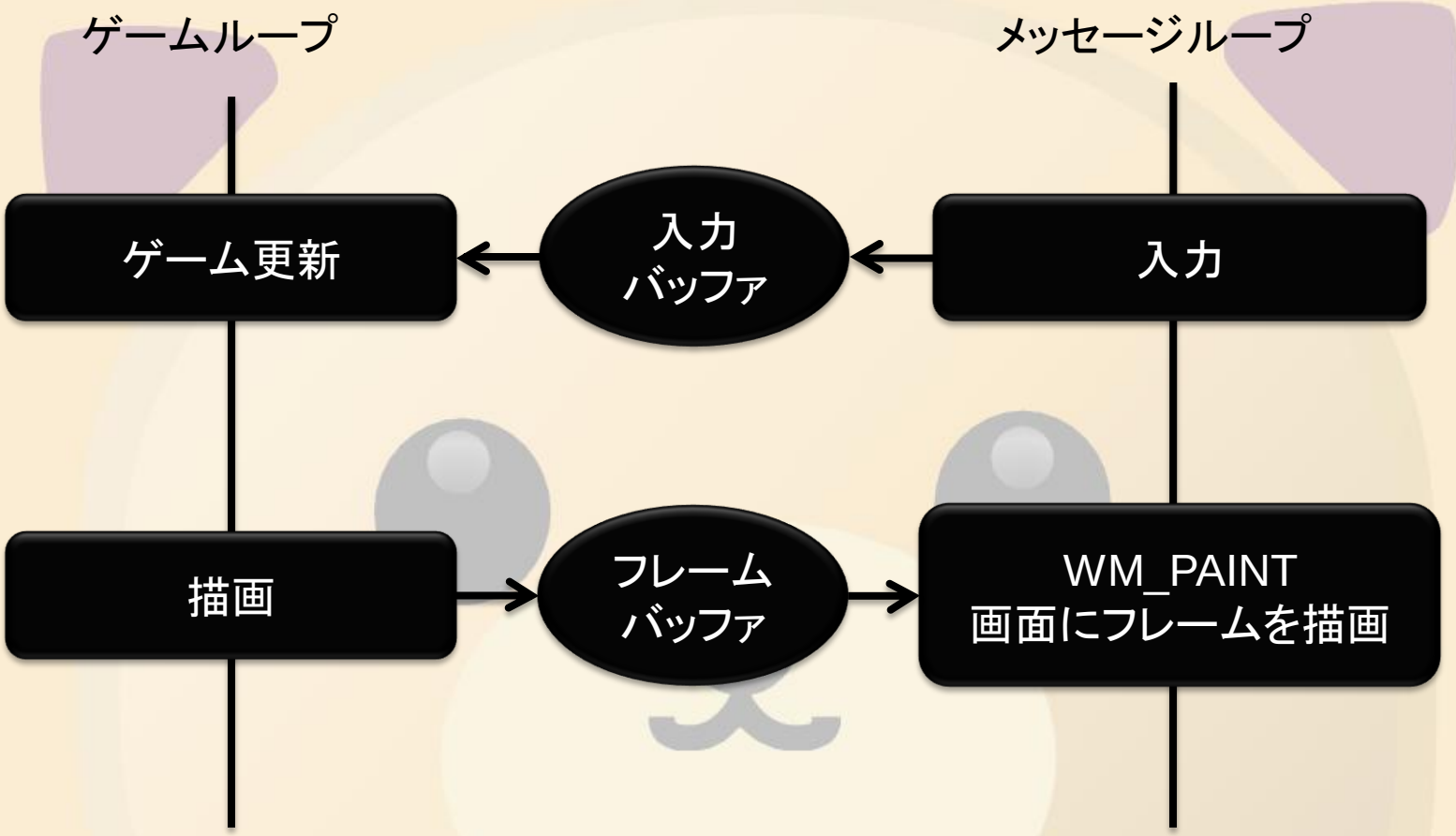
ゲームループ

- データ更新と描画の繰り返し
  - 1秒間に30～60回実行
  - フレーム数や時間の追跡
    - ゲームの進捗制御に必要
    - ハードウェアの性能に依存しない速度制御
    - ゲーム時間と実時間の分離
- プレゼンテーション処理の統合
  - 複雑な処理のパイプライン化


ゲーム起動
初期化処理
ゲームループ
データ更新
座標計算
あたり判定
通信
入力チェック
サウンドの更新
描画
ゲーム終了
リソースの解放

Windowsにおけるゲームループの実装

- メッセージループとは分離する
  - 別スレッドでの実装
  - アイドル時間を利用した実装
- メッセージの通知方法
  - メッセージの処理をバッファに記録
  - ゲームループからバッファを参照
- メッセージループとゲームループは通信しない
  - 必要な場合は同期オブジェクトなどを通す


ゲームループ
メッセージループ
ゲーム更新
入力
バッファ
入力
描画
フレーム
バッファ
WM_PAINT
画面にフレームを描画

ゲームで扱うリソース

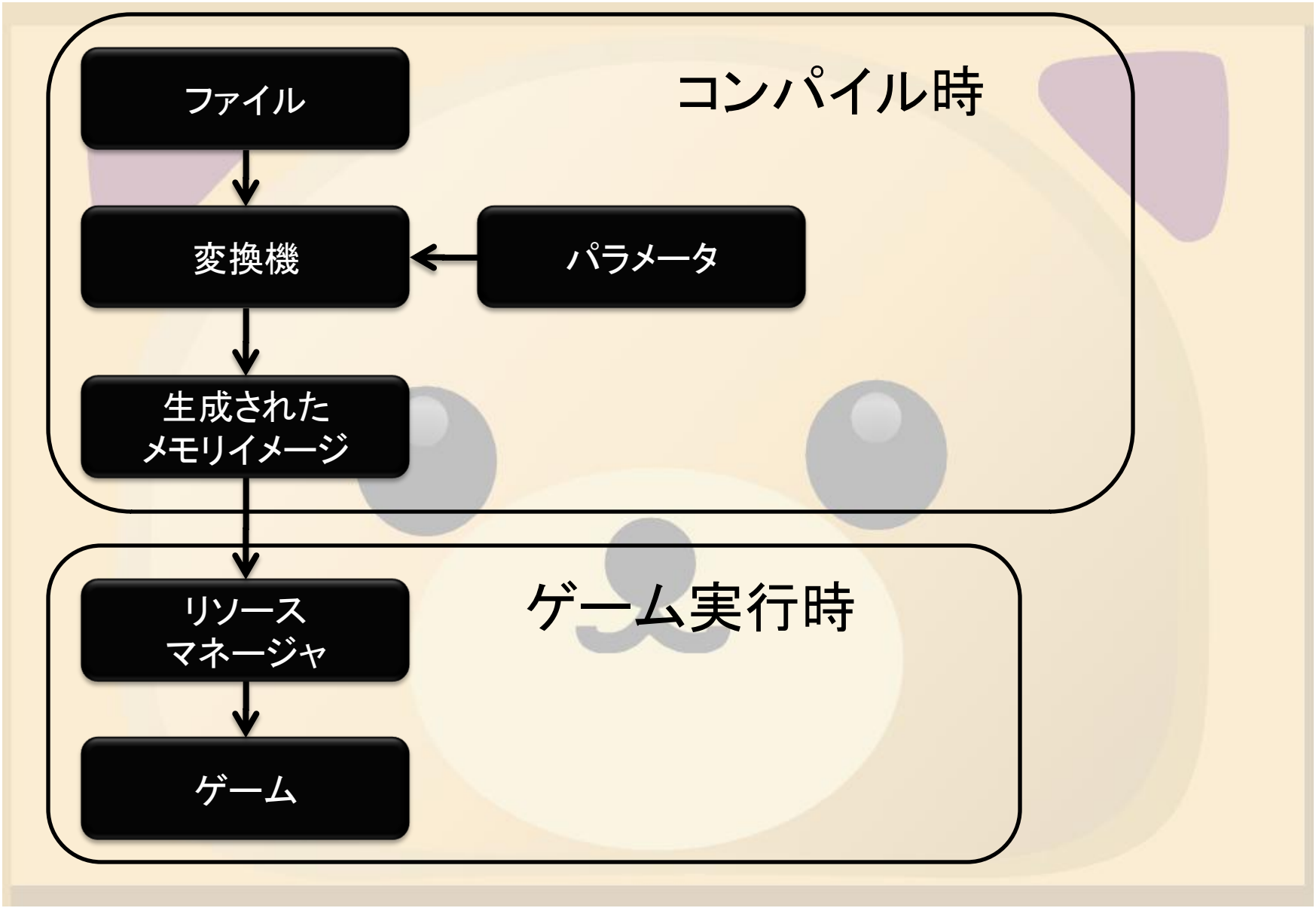
- イメージ
- フォント
- サウンド
- ゲームデータ
  - シナリオデータ
  - ステージデータ
  - キャラクターパラメータ
  - Etc…

リソース管理の課題

- 分業制・水平分散的開発
  - 絵はグラフィックデザイナーに
  - サウンドはサウンドデザイナーに
  - 声は声優に
  - シナリオはシナリオライターに
- 知的創造を妨げてはならない
  - 開発事情でデザインを制約しない

リソース管理方法

- リソースはコンパイル時処理
  - メタデータなど不要な情報を削除する
  - コンパイル時にメモリイメージに変換する
    - 実行時はバッファにファイルを読み込むだけ
    - 事前にリソースを処理できるため、仕様変更に強い
    - 暗号化などセキュリティの導入
- 実行時のリソース管理を統合
  - リソースの読み込みや解放・管理は一元化する


コンパイル時
ファイル
変換機
パラメータ
生成された
メモリイメージ
ゲーム実行時
リソース
マネージャ
ゲーム

変化する開発モデル

- ネットワーク・サービスとの結合
  - Game as a Service？
  - Game + service？
- 業務アプリケーションと同じような3層構造
  - サービスとの接続
  - 他のゲームインスタンスとの接続
  - リモート配置されたデータソース

プラットフォームの多様化

- マネージ・仮想環境上のゲーム
  - Java
  - Flash
  - .NET Framework
- 実行環境
  - モバイル
  - ブラウザ
  - PC・専用コンソール

まとめ

- 業務アプリケーション開発者とゲーム開発者の交流が重要
  - 業務ではUXについて参考になることが多い
  - ゲームではSOAやデータベース設計など、業務分野のノウハウを吸収するべき
- よりオープンなプラットフォーム化が必要
  - 多様化するデバイスへの対応
  - 教育や他業種との連携、参入障壁の排除