テストの視点で見たゲーム開発の流れと品質を支える仕組み

株式会社セガゲームス
開発技術部
粉川　貴至

自己紹介

自己紹介

- 粉川　貴至（こかわ　たかし）
- 株式会社セガゲームス所属
- CEDEC(Computer Entertainment Developers Conference)
  - 運営委員（2011～2015）
  - JaSST×CEDECコラボ企画担当（2011～2015）
- 書籍：「ゲームクリエイターが知るべき97のこと」に寄稿
  オライリージャパン、2012
- 自動化、QAエンジニアリングというトピックを中心に活動

@Kokawa_Takashi

CEDEC
(Computer Entertainment Developers Conference)

- 日本最大のコンピュータエンターテインメント開発者向けカンファレンス
- 2015年は6,373人の参加者。224セッション、展示ブース51
- 分野定義：エンジニアリング、プロダクション、ビジュアルアーツ、ビジネス＆プロデュース、サウンド、ゲームデザイン、アカデミック・基盤技術
- 資料のアーカイブはCEDiL（CEDEC Digital Library）に

https://cedil.cesa.or.jp/

JaSST×CEDECコラボ企画

- JaSST Tokyo, CEDEC それぞれで実施
- JaSST'11 Tokyo CEDECコラボ企画「ゲーム開発の世界から～金をドブに捨てないようにするテスト～」
  http://www.jasst.jp/archives/jasst11e.html#project5
- CEDEC2015 [JaSST×CEDECコラボセッション] 組み込みソフトウェアのシステムテスト自動化による作業の効率化
  https://cedil.cesa.or.jp/cedil_sessions/view/1380
- など

テストの視点で見たゲーム開発の流れ
と
品質を支える仕組み

※これから紹介する開発の流れや役割は一例で、会社や組織によって実態は異なります

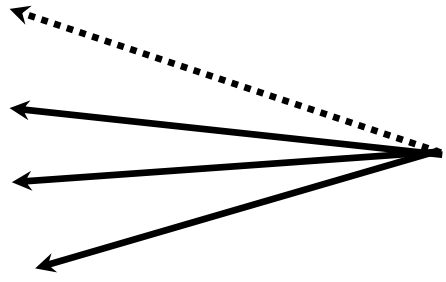
本講演での
ゲーム＝ビデオゲーム
を指すものとします

テストの視点で見たゲーム開発の流れ

ゲームの「テスト」というと
↓
デバッグ、プレイテストなどの
ゲームをプレイして動作を確認するテスト
が思い付き易いですが

ゲーム開発の流れ（マイルストーン）

- プリプロダクション
- プロダクション開始（予算、スケジュール、人員などが固まる）
- アルファ
- ベータ
- リリース
- アップデート



マイルストーン毎に
できあがったゲームを
プレイして動作を確認する

この部分以外の話をたくさんしていきます

ゲームシステムの特徴

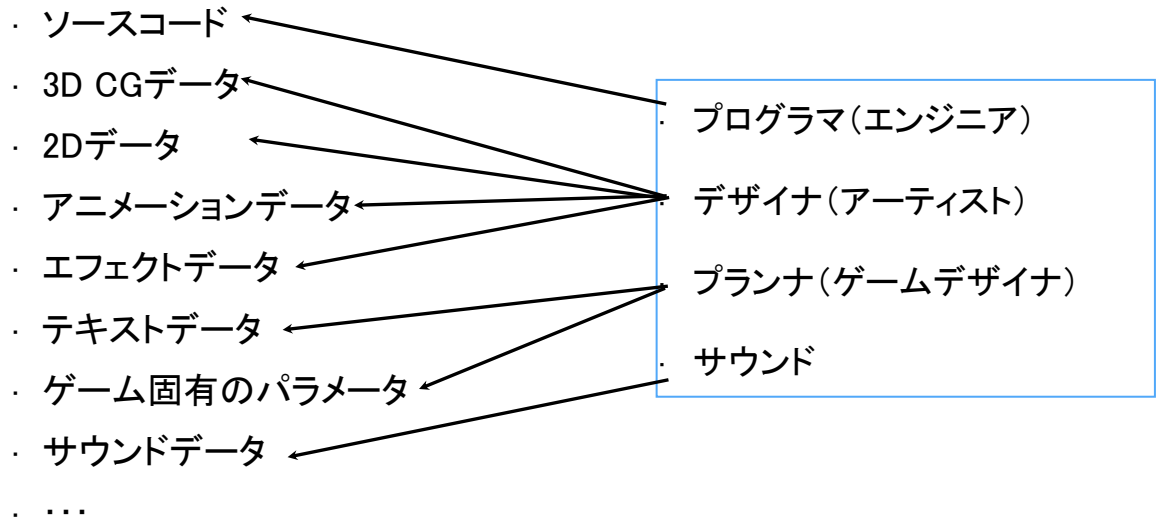
- 画面更新
  - 1秒間に30回、60回などの頻度で画面を更新する（フレームレート）
  - 次の更新タイミングまでに処理を完了させる。場合によっては分ける。
- 高速化、最適化が行われることが前提の部分
  - データ読み込み：データ構造の最適化、先読み、メモリに残す・流用
  - 3Dグラフィックス技術：GPUの使用、3Dデータ量のバランス、1フレームで実現できる表現の数
  - 衝突判定：真面目に全ての（形状の）判定はしない。ゲーム体験を損なわない境界線で実装。
  - ネットワーク：正しい事をチェックしないといけない箇所。遅延でゲーム体験が損なわれるとまずい箇所。

役割

- プロデューサー
- ディレクター
- プログラマ（エンジニア）
- デザイナ（アーティスト）
- サウンド
- プランナ（ゲームデザイナ）

ゲームに組み込まれる
コンテンツを作る人達

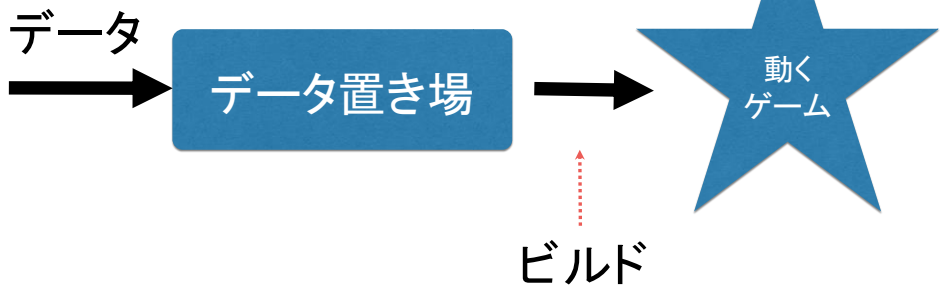
ゲームの構成要素



ソースコード以外の
データ（リソース）がとても多い

日々の開発作業

- プログラマ
- デザイナ
- プランナ
- サウンド

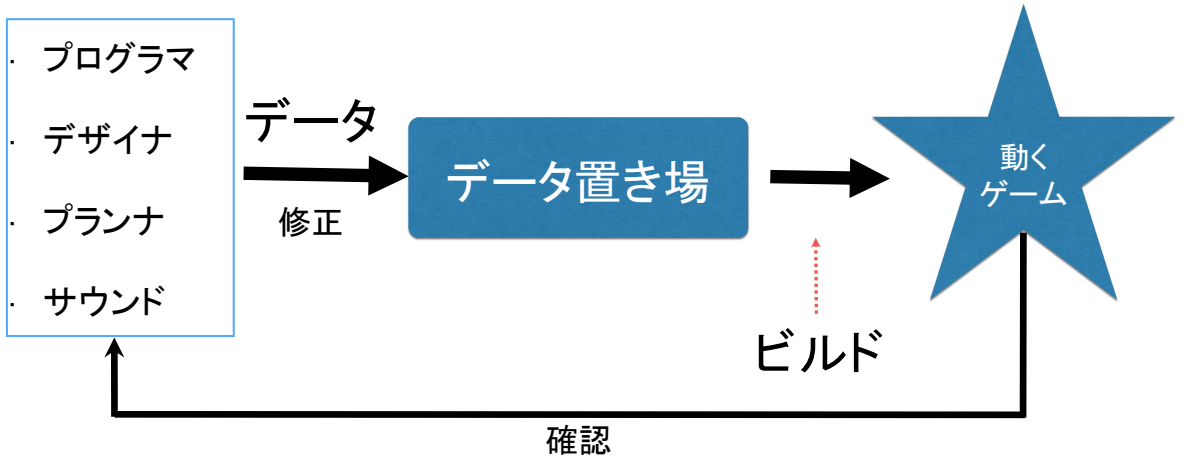


ビルド

ビルド

- ソースコードを実行ファイルに変換する
- 実行ファイル、バイナリデータ（後述）をパッケージ化する
  - 形式や呼び方は対象プラットフォームによって異なる

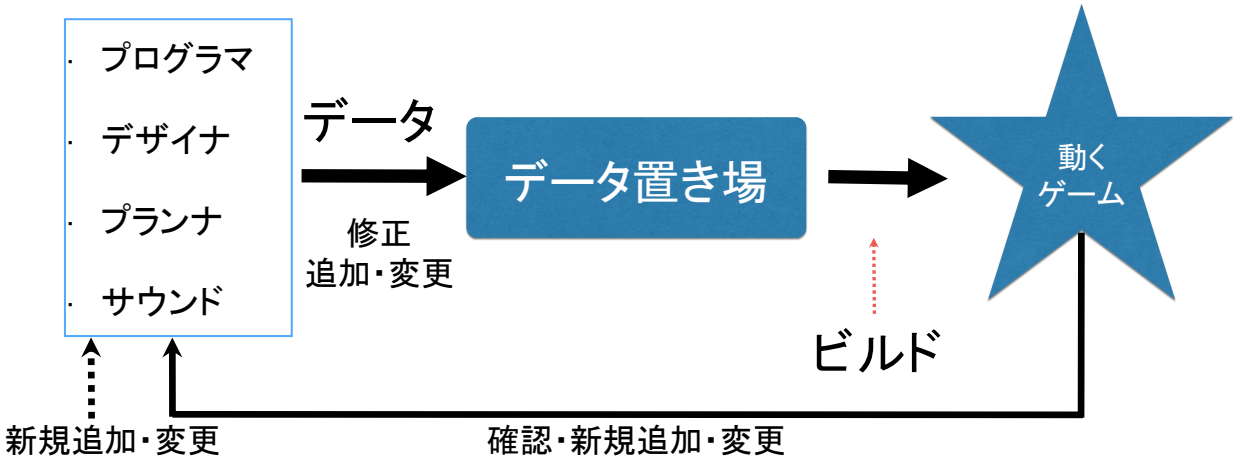
日々の開発作業
（バグ発見と修正だけなら）



データ作成の流れ

- プリプロダクションの段階で基本設計（試行錯誤）を行う
  - 基本となるゲームシステム
  - ポリゴン数、3Dグラフィックス技術（表現）、アニメーションを扱うためのデータ構造・・・
  - 少人数でそのゲームのキモになる最小限の目に見えるモノを作る
  - 仕様変更は日々発生する
- プロダクション以降
  - 基本設計をもとに、バリエーションや付加価値、製品化のために必要な機能を開発していく
  - プリプロダクション時には無かったが基本設計に追加しないといけない機能が発生したり、基本設計の変更が必要な要求が発生する事もよくある

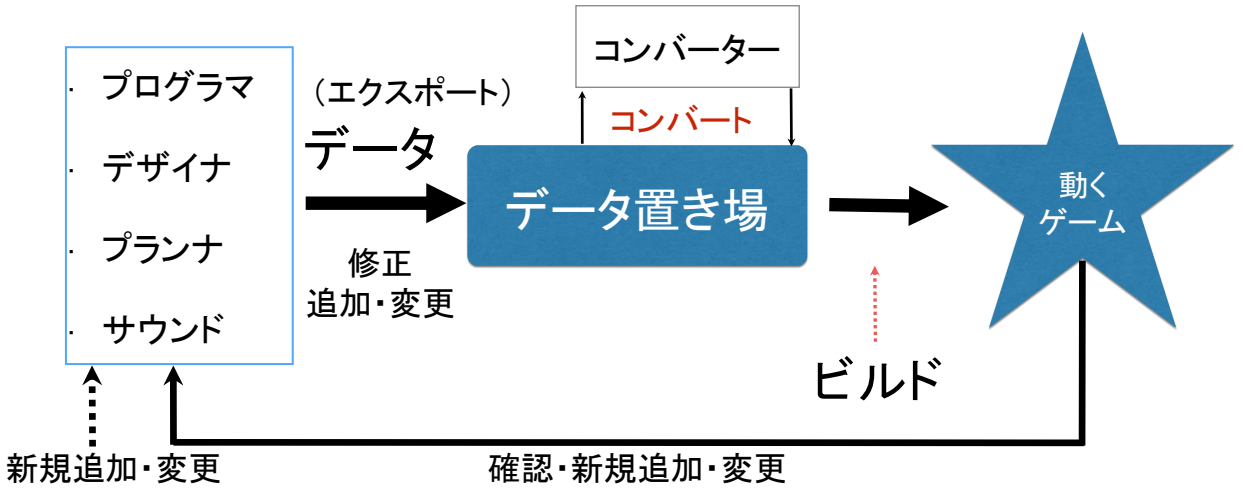
日々の開発作業



ゲームへのデータ組み込み

- データ作成ツールのデータ→中間データ→バイナリデータと変換される事が一般的
  - データ作成ツールのデータ：Maya, Photoshop, Excelなど
    - データサイズが大きい。ゲームに必要なデータを中間データに"エクスポート"する
  - 中間データ：3D CGデータの汎用フォーマットではFBXが有名、ゲームパラメータではJSON形式など
    - 必要なデータが周辺ツールなどから扱い易い構造になって保存されている事が多い。バイナリデータに"コンバート"して使う
  - バイナリデータ：独自形式である事が多い
    - ゲーム実行時に読み込んで使用するデータ。データサイズやセキュリティ対策などを考慮して最適化される

日々の開発作業



ゲーム開発の流れ
ここまで

問題が起こりそうなところ（不穏な空気）
が想像できますか？

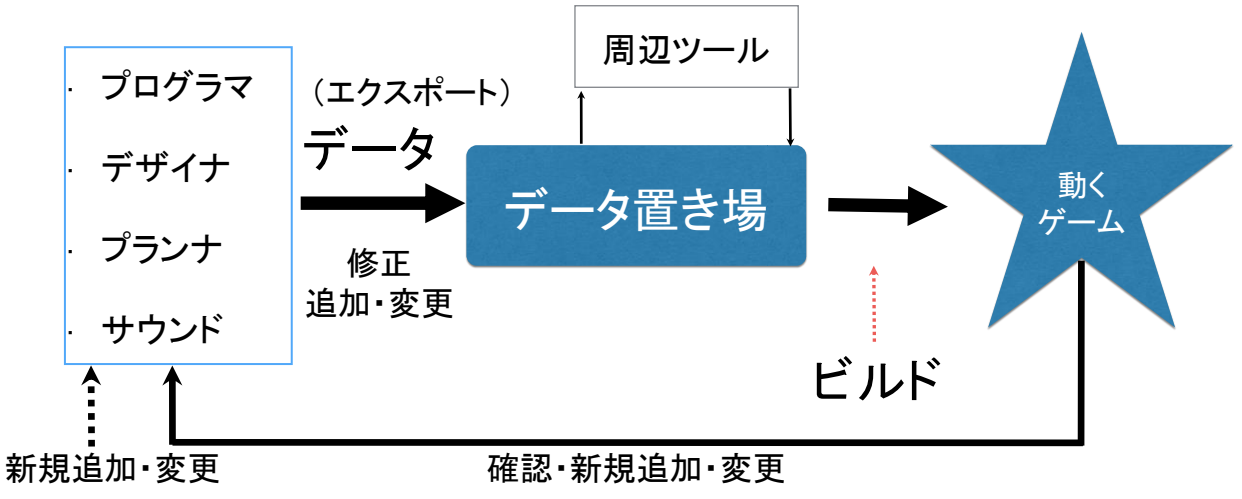
起こりそうな問題

起こりそうな問題（ヒント）

- ゲームを動かす前に起こる問題
- データのやり取り
- 複雑な手順
- 頻繁なデータ更新

- ゲームを動かす前に検出できる問題
  - 間違えないようにデータを置く、データが間違っていないか確認する、管理する
  - データ単体のチェックを入れる
  - ソースコードの品質チェックも先にできると良いですよね
- ゲームを動かしてデータが意図通りになっているか確認する
  - 作った人が確認する
  - 作った人以外が確認する ← プレイテスト

日々の開発作業

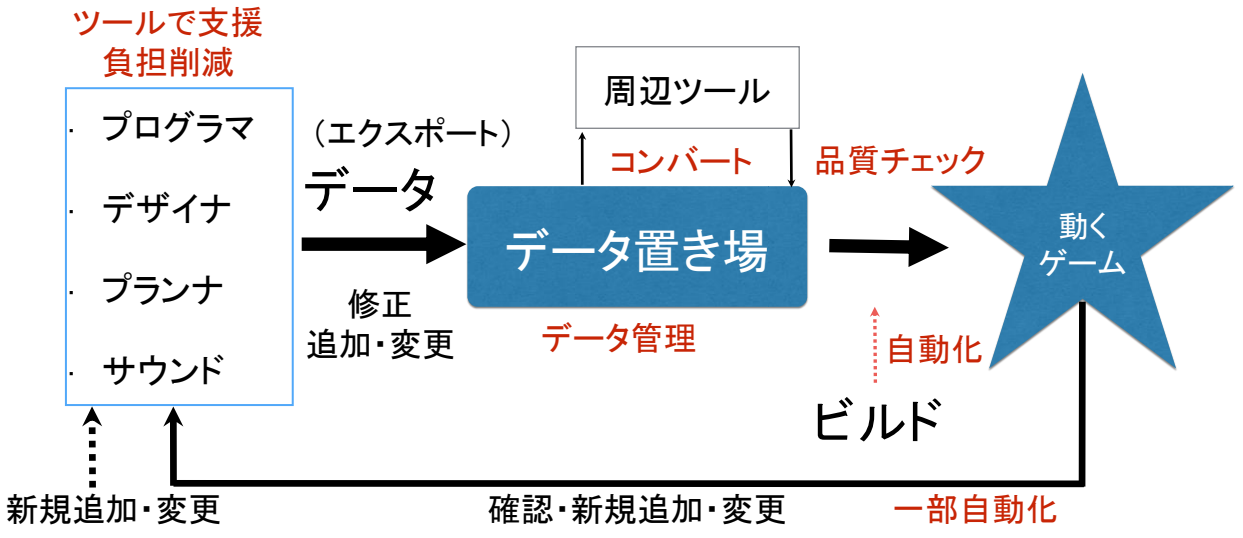


テストの視点から見ると
この部分の問題を（仕組みで）解決する事で
製品の品質が上がる

- クリエイターをクリエイティブな作業に集中させる
- テスターを雇っては？
  - 開発内テスター（エンベデッドテスター）も存在するケースもある
  - 動くゲームより手前の部分にはQAエンジニアが必要
    - 開発寄りのスキルセットが必要になる
  - 開発者がそのまま確認した方が早い、（短期的に）安い

このあたりに関わる仕事を普段しています

日々の開発作業



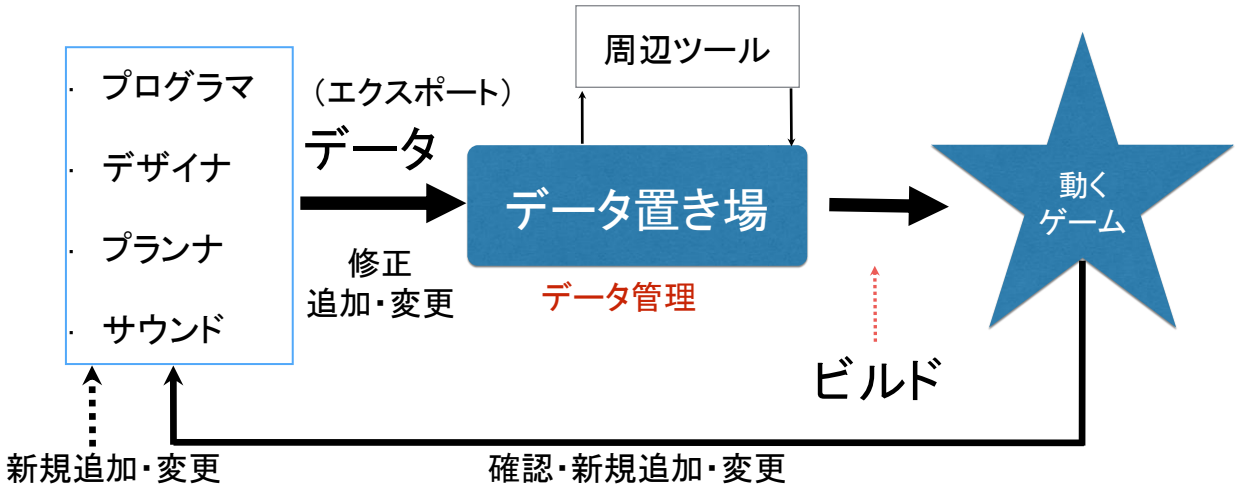
テストの視点で見たゲーム開発の流れ
と
品質を支える仕組み

品質を支える仕組み

品質を支える仕組み

- データ管理
- 自動コンバート、自動ビルド
- 自動デプロイ
- 自動テスト
- データの品質チェック
- バグトラッキングシステム

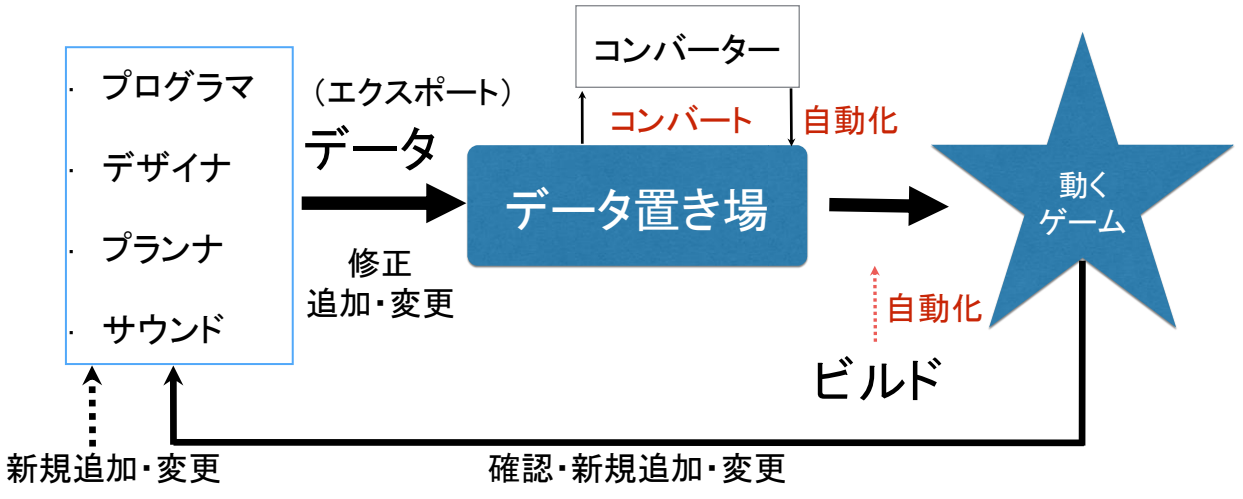
データ管理



データ管理

- バージョン管理ツール（SCM:ソフトウェア構成管理と呼ぶ事もある）を使う
  - Subversion, Git, Perforce...
  - プログラマ（エンジニア）以外のメンバーが使う事も想定しなければならない
- データ（数、量）が多い、更新頻度も高い
  - 数GB～数TB？
  - 管理するデータや方針を考えなくてはならない
    - ″デザインリソースのオリジナルは共有フォルダ、中間データをバージョン管理″など
  - ソースコードとそれ以外を分ける事も

自動コンバート、自動ビルド



自動コンバート

- コンバーターを自動で実行し、最新のコンバートデータをデータ置き場に置く
- コンバーターはゲームタイトル毎や組織毎に作られる事が多い

データ管理、自動コンバート事例

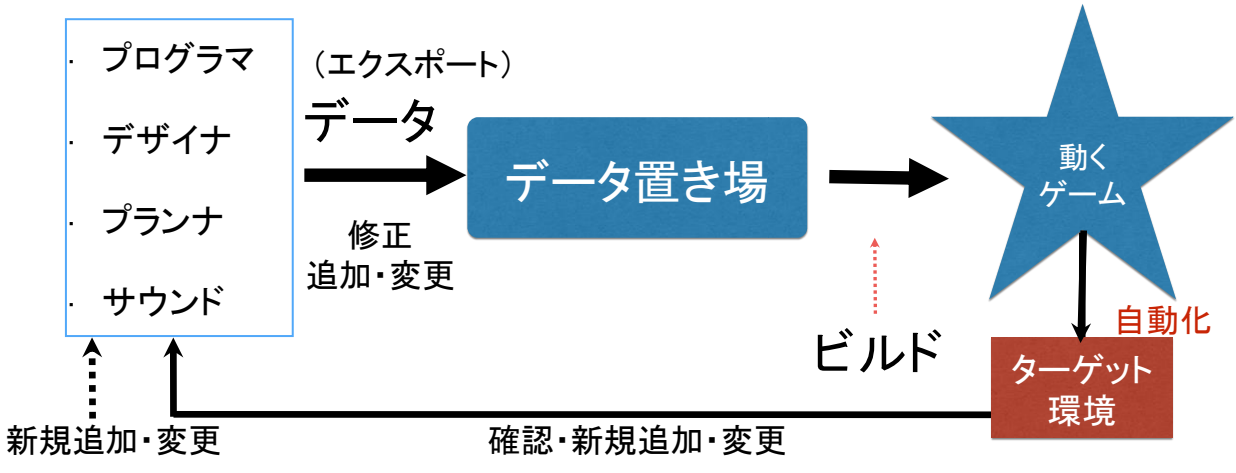
- CEDEC2015「長期運営タイトルに後からパイプラインの自動化を導入した際の技術的Tips」セガゲームス、粉川

https://cedil.cesa.or.jp/cedil_sessions/view/1306

自動ビルド

- データの更新状況を監視し、必要なタイミングで自動でビルドを行う
- ビルドツールを呼び出し、ソースコードのコンパイル、アプリケーション実行ファイルの作成、パッケージングなどを行う

自動デプロイ



自動デプロイ

- 実行ファイルを必要な場所に配置(デプロイ)し、動作確認できる状態にする
  - Webアプリケーションではwebサーバに配置する事を指す
  - ゲームではゲーム機やスマートフォンなど動作させる端末にインストールする(またはその直前まで準備する)事が該当する
    - 必要なデータがサーバにある場合、それらの配置も含まれる

自動化ツール

Jenkins

http://jenkins-ci.org/

- Jenkinsが有名
  - オープンソースソフトウェア
  - シンプルかつ必要な機能をプラグインで拡張可能
  - Web画面で全ての操作ができる
  - 大事な機能：スケジューリングと通知

継続的インテグレーション（CI）

- アジャイル開発
  - XP（エクストリーム・プログラミング）
    - 「コミュニケーション・シンプルさ・フィードバック・勇気」を提唱
      具体的な19の実践（プラクティス）がある（テスト駆動開発、ペアプログラミング、リファクタリングなど）
      - 継続的インテグレーション（Continuous Integration）
        - XPの中でも効果的な実践の1つ：頻繁にビルド・テスト・結合を繰り返す

自分なりの継続的インテグレーションの受け止め方

- 技術と技術の間を繋ぐ
- ツールとツールの間を繋ぐ
- 作業と作業の間を繋ぐ
- 開発の流れを出来るだけ止めないように自動化して繋ぐ
- ここまで紹介した仕組みも、この後紹介する仕組みも繋いで継続的インテグレーションで回せるようにしています

自動テスト

プレイテストの自動化

- ゲームでは難しい。なぜか
  - 汎用的な"テストハーネス"が存在しない
    - WebでのSeleniumのような
  - 入力の自動化機構やツールは存在するが、出力検証の仕組みが難しい
    - Webやスマホで言うアプリ外のテストツールから情報取得可能なUIコンポーネントが無い
    - 出力画面の画像比較は可能だが、効果の出るケースが限定される
  - 大きいプロジェクトではそのゲームに特化した形で機能が実装されている事はある

ゲームエンジンの登場

- Unityや、Unreal Engine 4といったゲームエンジンが多く使われるようになってきている
- エンジンがテストハーネスを用意しているケースでは、その仕組みに則った自動テストが可能
  - 特定のステージ（シーン）を読み込む⇒決められた手順でゲーム内を操作する⇒結果の状態が正しいか（想定通りか）を確認する

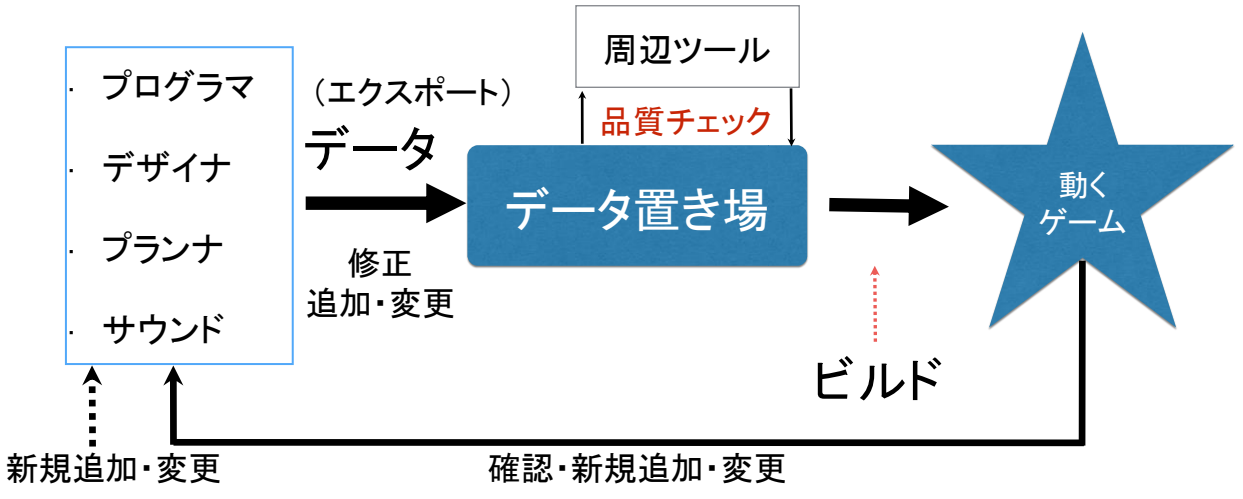
プレイテストの自動化事例

- 「Jenkinsを使ったコンシューマゲームでのデプロイとテスト」
  株式会社イリンクス,Jenkins ユーザ・カンファレンス 2015 東京
  - http://www.slideshare.net/swiftnest/jenkins-43394510
- 「10ヵ月でHDゲームを開発する方法 ～龍が如くを支えたテクノロジ～」
  株式会社セガ（当時）, CEDEC2010
  - https://cedil.cesa.or.jp/cedil_sessions/view/324
- 「バグチェック作業の自動化について」
  Bayonetta 2 開発者ブログ
  - http://www.platinumgames.co.jp/bayonetta2/archives/881

単体テストについて

- 実装言語に対応しているユニットテストフレームワークを使う
  - C++：CppUnit や GoogleTest など
- ″変更が多い箇所ではメンテナンスコストが高くなるので運用できない″という点がよく聞く課題

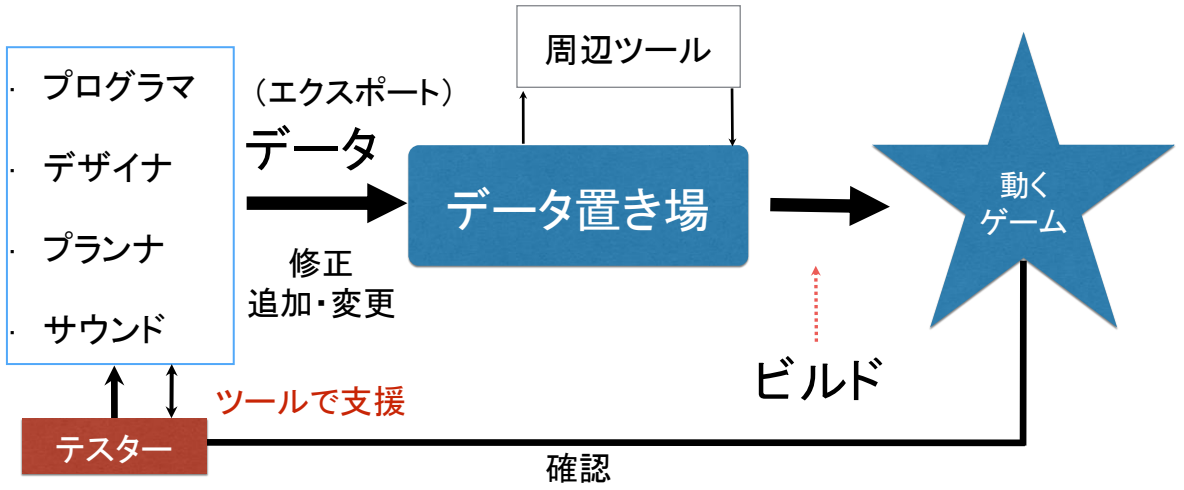
データの品質チェック



データの品質チェック

- ソースコード
  - 静的解析ツール
    - ソースコードを解析し、不具合を検出する
  - コンパイラ警告
  - コーディングルール
  - セキュリティや依存ライブラリのライセンスに問題が無いかのチェック
- リソースのチェック
  - コンバーターに含める事が多い
  - データ容量やフォーマットが正しいかどうかのチェック

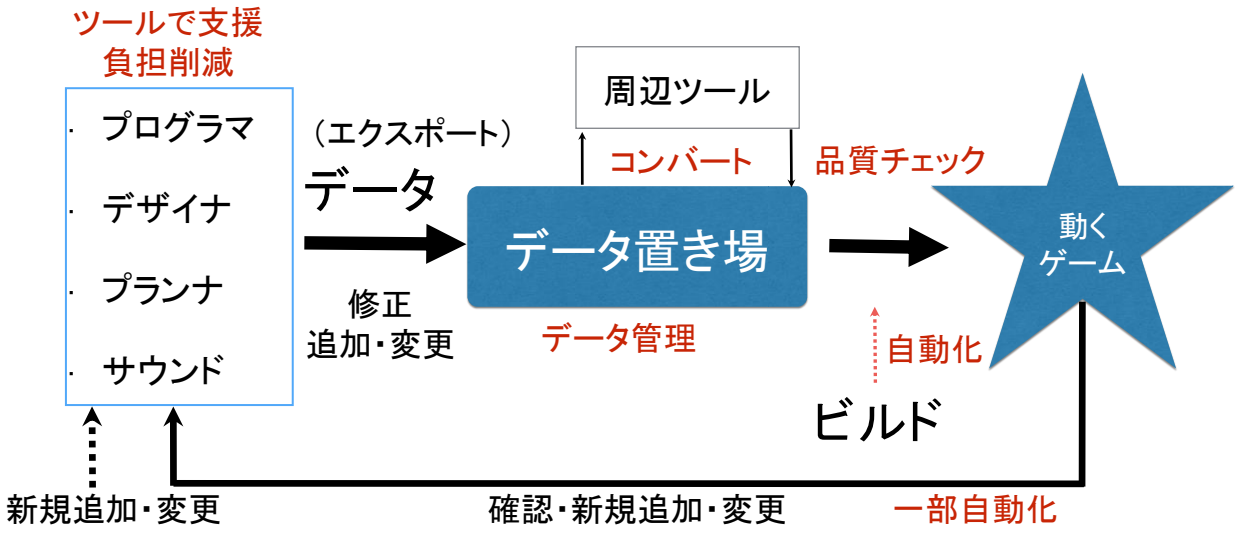
バグトラッキングシステム



バグトラッキングシステム(BTS)

- Redmine, Trac, Mantis, JIRA など
- バグ1つ1つに対して、"チケット"を作成して管理する
  - "いつ", "どこで", "度のバージョンで", "どのようにしたら", "どうおかしくなったのか"など必要な情報をシステムの項目として入力可能に
  - "(テスターが)バグ発生報告"⇒"(開発者が)修正対応"⇒"(テスターが)修正確認"⇒"終了"といったワークフローをシステムとして定義する
  - コミュニケーションミスによる修正漏れを防ぐ

日々の開発作業



こういった仕組みなどで
日々の開発作業における品質は
支えられています