オンラインゲームの基礎知識

株式会社セガ
技術開発センター ネットワークセクション
阿形　佳美

はじめに

- オンラインゲームの特徴をわかりやすくするため、ゲーム専用機に特化して説明します。
- ゲーム機の仕組み、スタンドアローンのゲーム開発手法を説明します。
- オンライン化することでどういった問題が発生してくるか説明します。

ゲーム専用機

コンシューマー	アーケード
Dreamcast	NAOMI, NAOMI 2
PS2	System246
GAME CUBE	TriForce
Xbox	Chihiro

ゲーム機の思想

コストダウン	比較的貧弱なCPU 少ないメモリ容量 ハードディスク無し
ハードウェア性能を 最大限に使う	OS無し、あるいは最低限の機能のみ
画面表示重視	強力なグラフィック性能と、 それを最大限に活かす実装

「貧弱なCPU性能、メモリ容量」への対策

- OS無し or 最低限の機能のみ
- OSのオーバーヘッドの解消
- OS分のメモリ容量を節約
- CPU、メモリ(キャッシュ等も含む)を最大限使えるようにアプリケーションでスケジューリングし最適化

最適化の代償

割り込みが使いにくい！

割り込みが入ると、折角最適化したのが無駄になる

タイマー割り込みも使いにくい

TCP、DHCP、PPP、DNSなど沢山あるタイマーをどうする？

入力処理どうする？

Unixなんかだと割り込みで実現してるみたいだけど…

プロトコル実装するの面倒すぎ！

割り込み対策

- しかたないので…
  - 入力はバッファにためておく
  - アプリケーション側から受信要求し、バッファからコピー
  - タイマーはV blankの回数を数える

「ハードディスクがない」

- 最近は付くようになりましたが
- 基本はCD-ROMなどの光ディスクメディア
  - 一度プレスしたら、変更するには再度プレスして交換するしかないので、非常にコストがかかる
- あると便利なんだけど…

プロトコルスタックの修正

プロトコルスタックはアプリケーションに
組み込んでいるので…

アプリケーションの修正

ROMの再プレス、配布

再プレスと配布費用がかさむから、やーめた！

修正しなくてすむように

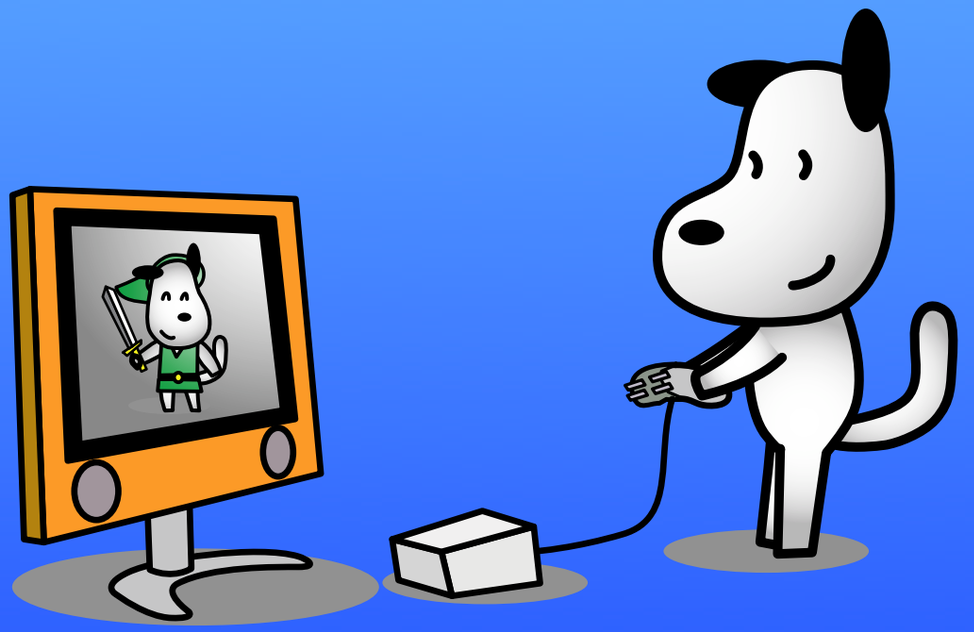
- 実環境でのテストを十分に行う
  - それでも問題が出るときは出る
- 新しいプロトコルにはどっちにしろ対応できない

画面表示問題

- ゲーム機において、なぜそんなに画面が重要なのかを説明します。
- そのためにとっている開発手法と、それによってオンラインゲームでどのような問題が発生してくるか説明します。

まず画面ありき

- ほとんどのゲームは画面中のプレイヤーキャラクタを動かすことで成立
- プレイヤーキャラクタが画面に出なくても、何らかの情報が画面に表示され、それをもとにゲームを進める
- よって、画面がちゃんと出ないと話にならない



ちゃんとした画面の定義

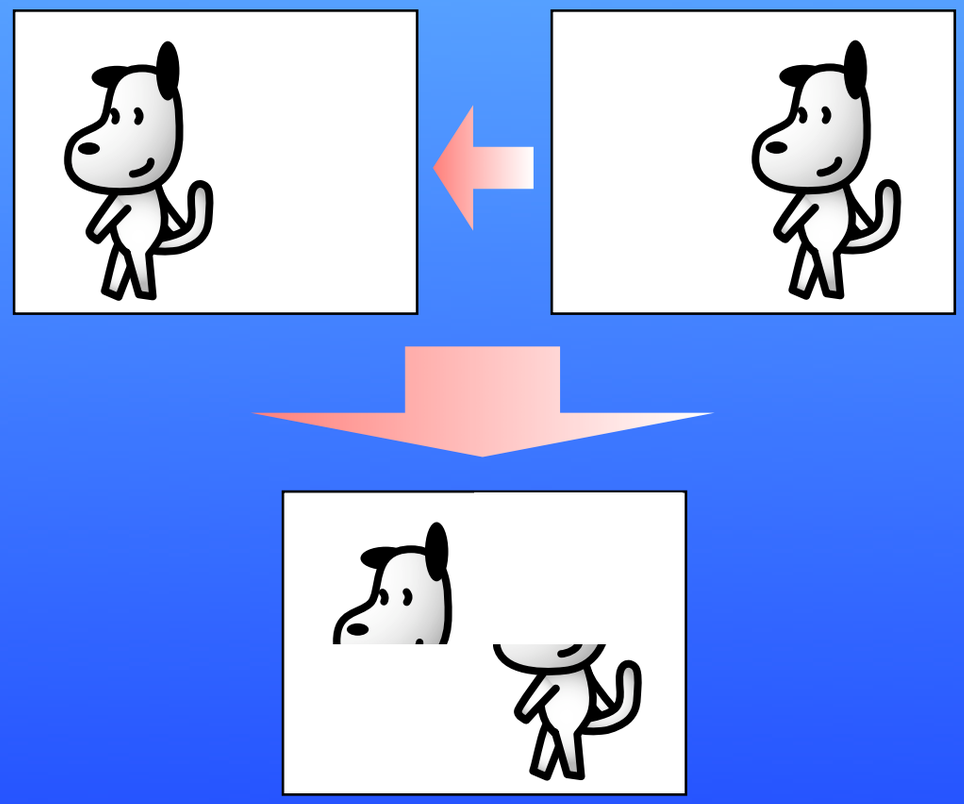
- ユーザーの操作に遅れることなく確実に反応が画面に反映されること
- コマ落ちなくスムーズに動くこと
- 家庭用TVで綺麗に見えること

家庭用TV

- NTSCフォーマットの場合
  - 約30fps インタレースで約60fps
  - ゲーム機の場合、これに同期して画面を更新する

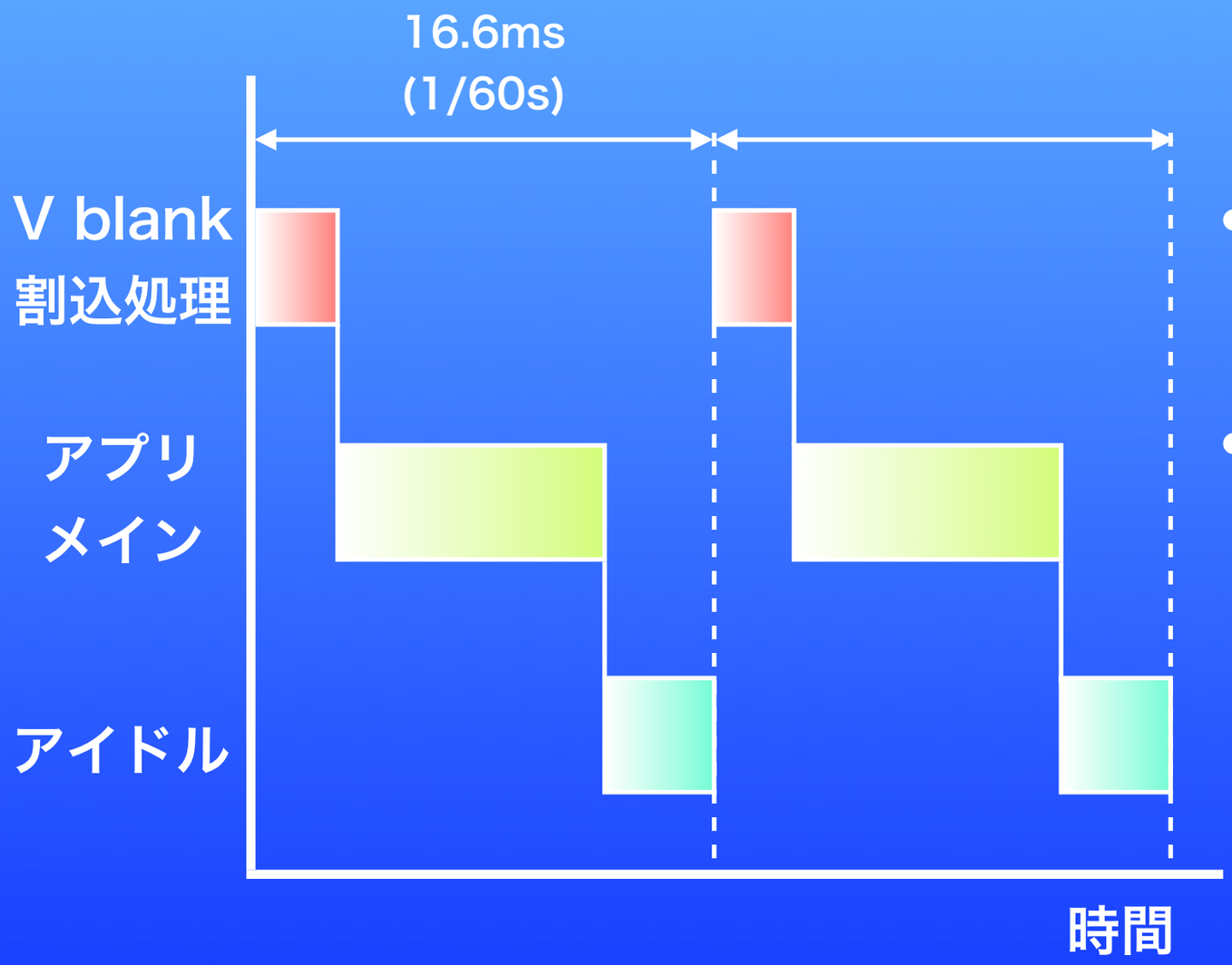
画面の更新

- なぜ同期しなければならないか？
- 同期せずに更新すると、書き換え途中の状態が表示されてしまう



書き換え途中の画面

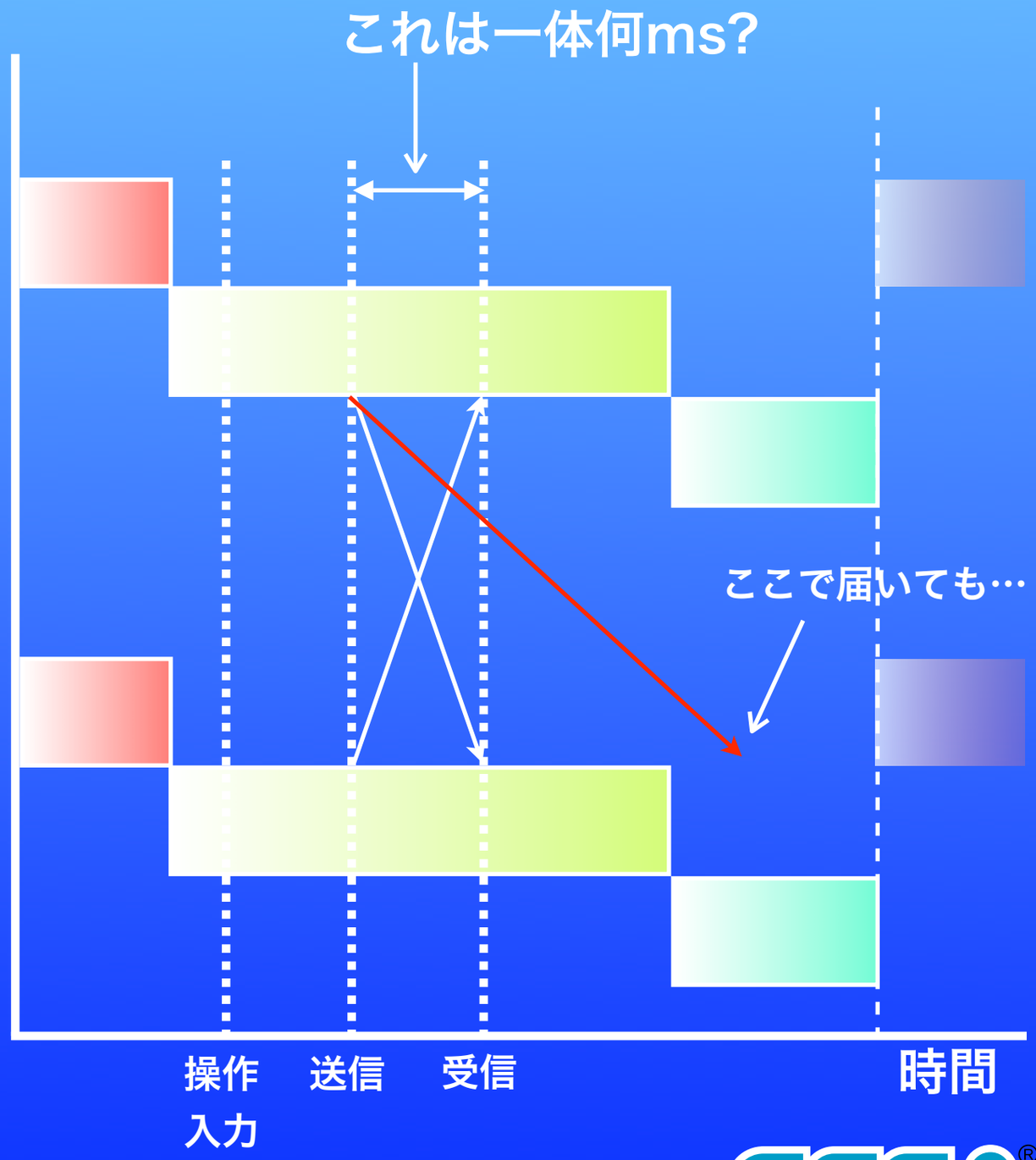
同期するために



- V blank割り込みをベースにタスクを組み立てる
- 1 V blank 周期の間にプレイヤーの操作情報を処理して、次の画面を用意

オンラインゲームの場合

- これまで通り、V blankを基準にすると
  - 遅延
    - 非常に短い時間(少なくとも16ms以下)で届く必要がある
- パケットロスや到着順序の入れ替わり
  - あってはならない



現実には不可能なので…

- なんらかの対策が必要
- せめて数百msぐらいまで許容できないと、現実的でない
- どうにかして「ごまかす」しかない
  - ゲームのルール自体で工夫
  - データのやりとりで工夫

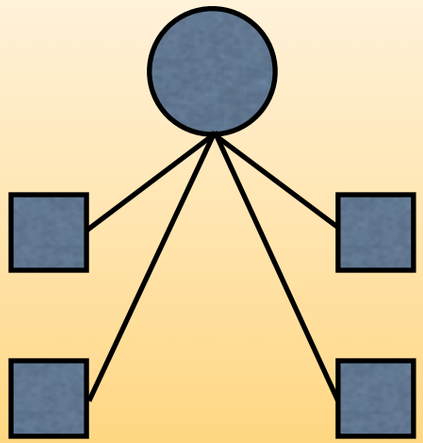
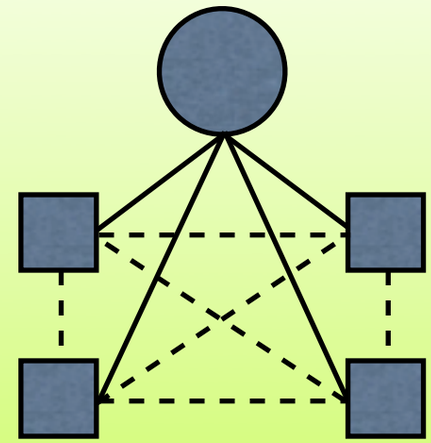
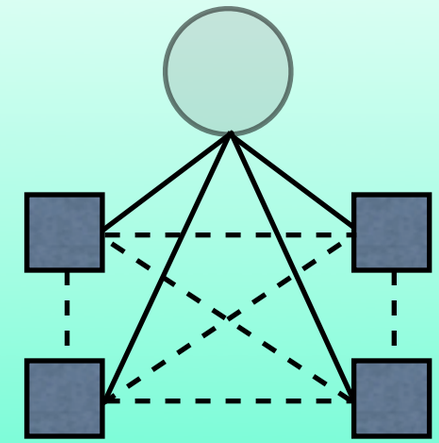
ゲームのルールで工夫

- 遅延が大きくても問題の無いルール作り
- 相手行動を待つタイプにする(ターン制)
- ローカルの情報をバッファして、あとで判定
- プレイヤー同士のかかわり合いを減らす

データのやり取りで工夫

- データを分類し、内容に応じた処理方法を選択する
  - プロトコル
  - トポロジー
  - etc…

トポロジー

	スター型	ハイブリッド型	メッシュ(P2P)型
通信形態	全てサーバー経由で通信する。 	スター型、メッシュ型の混合 	端末同士で直接通信する。 
メリット	情報の整合性をチェックできる。	スター型、メッシュ型それぞれの長所を活かせる。	遅延について、スター型より有利。

遅延耐性と更新頻度による分類

小

大

遅延

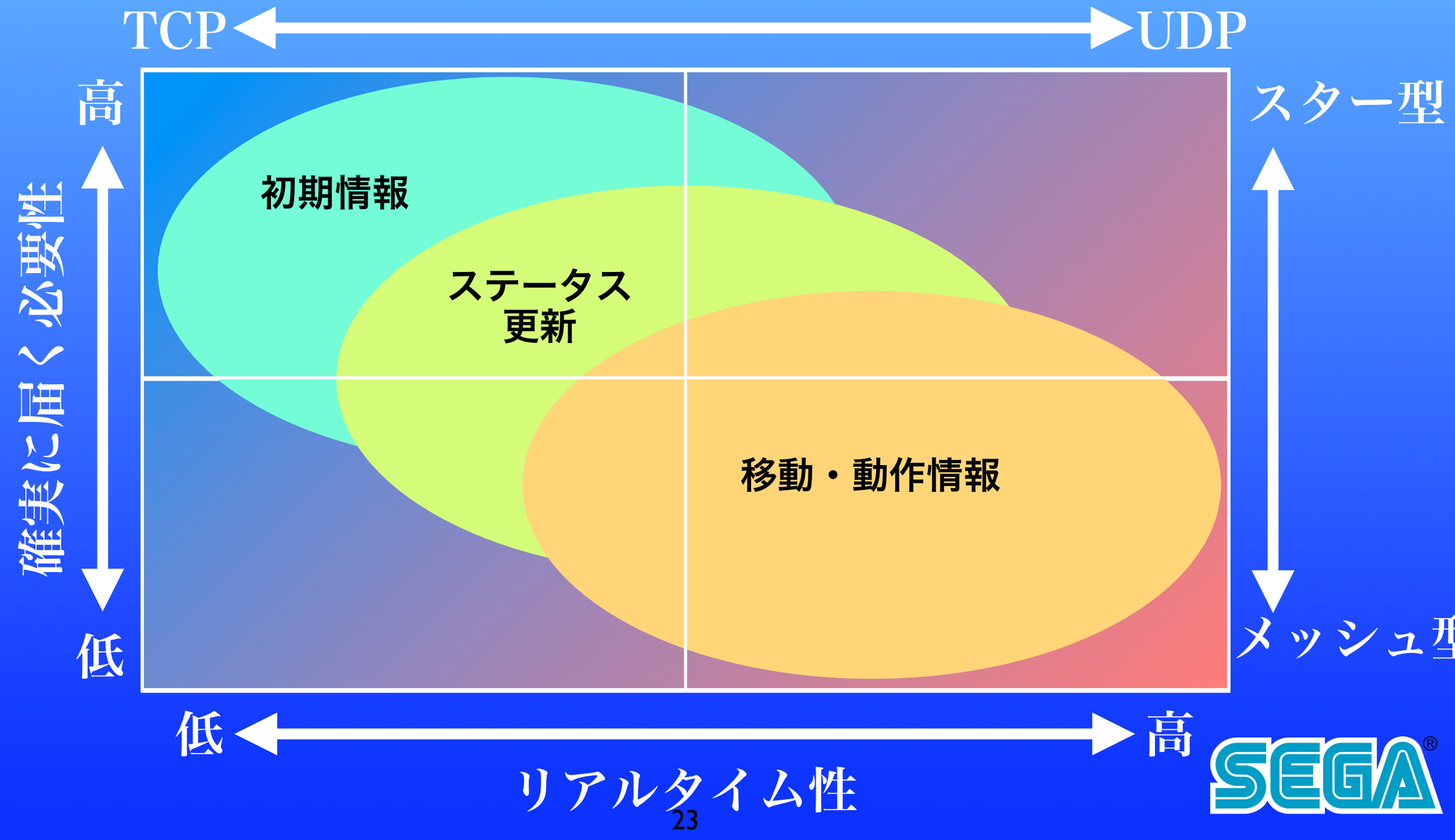
多

少

更新

移動、動作情報	キャラクターの移動 キャラクターの動作情報
ステータス更新	フィールドのステータス変化 キャラクターのステータス変化 アイテムのステータス変化
初期情報	フィールドの地形とステータス キャラクターの位置とステータス アイテムの位置とステータス

情報の種類と
トポロジー、プロトコル



実際のトポロジーとプロトコル

- スター型＋TCP（クライアント側から接続）という選択をせざるを得ない面が多い
- NAT対策
- 遅延にシビアなゲームは、可能であれば、P2PやUDPにしたいけど…

まとめ(?)

- Internetのプロトコルはゲーム機に優しくない
- 遅延耐性はゲームのルールにより大きく変わる
- ルールや実装を工夫してなんとか遅延をごまかしている
  - しかし、実装の工夫も環境による制限(NATなど)をかなり受けている

おしまい

ご清聴ありがとうございました。