□フィールド　リソース、データ最適化について

リソース、データの最適化により、

処理負荷を大きく改善することが可能となります。

今後発生する、リソース、データの最適化に向け、

・どういったリソースの修正が効果的なのか？　と

・何の処理負荷が高い状態かを知るためのパフォーマンスメータについて説明いたします。

　●フィールド内の高負荷処理とその対策方法

　１．高さチェック （マップモデラー）

　　低コスト化：コリジョンモデルのポリゴン数削減

（ポリゴンが密集している箇所のポリゴンをけずる）

目安として、1回の高さ取得でチェックするポリゴン数を１０程度まで落としたい。

パフォーマンス表示　影響範囲：collision player update mmodel update

　　デバッグ機能：field/DrawSwitch/コリジョン描画[1]



　２．マップ上で移動しているNPC （プランナー）

　　低コスト化：高さ取得をしない動作コードに変更する。（その場から移動しないもの）

　　パフォーマンス表示　影響範囲：motion mmodel update

　３．モーションの更新（プランナー）

　　低コスト化：同時に動く動作モデルの数を制限する。

　　　　　　　 ：可視カリングで、範囲外になるようなカメラアングルに変更する。

　　パフォーマンス表示　影響範囲：motion

　４．モデル、マテリアル数（デザイナー）

　　低コスト化：全体のマテリアル数の削減

　　パフォーマンス表示　影響範囲： render mesh

　５．レイアウトアニメーションの更新処理（UI）

　　低コスト化：表示した無くてよいペインを非表示にする。

　　パフォーマンス表示　影響範囲：layout upper layout lower

　６．天気のパーティクル（デザイナー）

　　低コスト化：パーティクル数を今の1/3程度に落とせないか？

　　パフォーマンス表示　影響範囲：scene calc render mesh

●PERFORMANCEメーターの説明



デバッグメニューの処理インフォ表示を２にするとPERFORMANCEメーターが表示されます。

　→field

　　→DrawSwitch

　　　→処理インフォ表示[2]

x/y us という表示について

xは今かかっている時間 yは、処理負荷オーバーしない状態のときの平均的な時間

　となっています。

各項目の説明

* Motion
  + モーションの更新にかかっている時間
* Motion num
  + モーションを更新しているNPCの数
* Collision
  + 自機が１回高さ判定するのに必要な時間
* Collision tri num
  + 自機が１回の高さ判定でチェックする三角形ポリゴンの数
* Render mesh
  + メッシュのレンダリングに必要な時間
* Mesh num
  + レンダリングしているメッシュの総数
* Layout Upper
  + 上画面のレイアウト描画に必要な時間
* Layout Upper Num
  + 上画面のレイアウト数
* Layout Lower
  + 下画面のレイアウト描画に必要な時間
* Layout Lower Num
  + 下画面のレイアウト数
* Player Update
  + 自機の移動チェックに必要な時間
* Mmodel　Update
  + NPCの更新に必要な時間
* EventCheck
  + イベントチェックにかかっている時間
* Gimmick Update
  + ギミックの更新にかかっている時間
* Scene　Calc
  + ３Dモデル、パーティクルの情報更新に必要な時間
* Scene　Submit
  + ３Dモデル、パーティクルのサブミットビューに必要な時間
* SubWin　Draw
  + 下画面独自の描画処理に必要な時間

以上となります。