# Strong Types

Following Jonathan Boccara’s concept on <https://www.fluentcpp.com/2016/12/08/strong-types-for-strong-interfaces/> I used strong types for GRID\_COORD and PIXEL.

# Coordinate systems

Windows coordinate system has origin at **top** left. See POINT, RECT

Direct3D has origin **bottom** left.

Evolution follows Direct3D philisophy wherever possible.

Grid coordinate system has origin **bottom** left. See GridPoint, GridRect

Pixel coordinate system has origin **bottom** left. See PixelPoint, PixelRect

Transformation between top-left and bottem-left coordinate systems happens at a low level close to the windows api.

* Pixel coordinates ( y = clientAreaHeight – y )
* Client area rectangle ( swap top/bottom )

# Control flow of user interactions

## Zoom factor in main grid window

Model:

* PixelCoordinates.m\_sFieldSize

Sources of action:

* „Fit“ in grid window context menue
* „Fit“ button in status bar
* „+“ and „-“ buttons
* „Zoom in“, „Zoom out“ in menue
* Mouse wheel in GridWindow
* Trackbar in status bar

Views:

* Main grid window
* Trackbar in status bar

## Simulation speed (delay between generations)

Model:

* PerformanceWindow.m\_dwGeneraitionDelay

Sources of action:

* MaxSpeed button in status bar
* Trackbar in status bar

Views:

* PerformanceWindow
* Trackbar in status bar

## Brush size

Model:

* Editorstate.m\_brushSize

Sources of action:

* History replay
* Script
* Trackbar in editor window

Views:

* Trackbar in editor window

## Brush intensity

Model:

* Editorstate.m\_usBrushIntensity

Sources of action:

* History replay
* Script
* Trackbar in editor window

Views:

* Trackbar in editor window

## Brush mode

Model:

* Editorstate.m\_brushMode

Sources of action:

* History replay
* Script
* Radio button group in editor window

Views:

* Radio button group in editor window

## Simulation/Edit mode

Model:

* Editorstate.m\_bSimulationMode

Sources of action:

* History replay
* Script
* Button in StatusBar

Views:

* Window visibilty (Performance/Editor)

## General strategy

Channel action events from all sources to central class which knows views and controllers and routes events to views, not to controllers.

Best candidate: EvoController

## Affected classes

* History Replay ???? (Source)
* Scripting ??? (Source)
* GridWindow (Source + View)
* StatusBar (Source + View)
* EvoController (Source)
* PixelCoordinates (Model)
* PerformanceWindow (Model + View)
* EditorWindow (Source + View)
* EditorState (Model)

# Control flow

## Normal case (no history replay, no script replay)

### Brush Mode

Main Thread

* [L8] User action: Radio Button in EditorWIndow
* [L8] EditorWindow:Userproc – WM\_COMMAND, IDM\_xxx
* [L7] WorkThreadInterface:PostSetxxx
* [L7] WorkThread:WorkMessage – THREAD\_MSG\_SET\_xxx
* [L7] WorkThread:postMessage – THREAD\_MSG\_SET\_xxx
* [L7] PostThreadMessage – THREAD\_MSG\_SET\_xxx

Worker Thread

* [L6] WorkThread:dispatchMessage – THREAD\_MSG\_SET\_xxx
* [L6] WorkThread::editorCommand( tEvoCmd::editSetxxx
* [L5] EvoHistorySys::EvoCreateEditorCommand( tEvoCmd::editSetxxx
* [L4] HistorySystemImpl::CreateAppCommand(( tEvoCmd::editSetxxx
* [L4] HistorySystemImpl::step2NextGeneration( tEvoCmd::editSetxxx
* [L3] EvoModelData::OnAppCommand( …
* [L2] EvolutionModelDataImpl::Setxxx
* [L1] EditorState::Setxxx

### Edit action (mouse click in grid window)

Main Thread

* [L8] User action: Mouse click in GridWindow
* [L8] GridWindow: Userproc – WM\_LBUTTONDOWN
* [L8] GridWindow::onMouseMove
* [L7] WorkThreadInterface:PostDoEdit
* [L7] WorkThread:WorkMessage
* [L7] WorkThread:postMessage
* [L7] PostThreadMessage

Worker Thread

* [L6] WorkThread:dispatchMessage – THREAD\_MSG\_DO\_EDIT
* [L6] WorkThread::editorCommand( tEvoCmd::editSetxxx
* [L5] EvoHistorySys::EvoCreateEditorCommand( tEvoCmd::editDoEdit
* [L4] HistorySystemImpl::CreateAppCommand(( tEvoCmd:: editDoEdit
* [L4] HistorySystemImpl::step2NextGeneration( tEvoCmd:: editDoEdit
* [L3] EvoModelData::OnAppCommand( …
* [L2] EvolutionModelDataImpl:: ModelDoEdit

### Single step

Main Thread

* [L8] User action: Click “SingleStep” button in StatusBar
* [L8] StatusBar: Userproc – WM\_COMMAND, IDM\_GENERATION
* [L8] AppWindow: Userproc – WM\_COMMAND, IDM\_GENERATION
* [L8] EvoController::ProcessCommand( IDM\_GENERATION
* [L7] WorkThreadInterface:PostGenerationStep
* [L7] WorkThreadInterface:postGotoGeneration
* [L7] WorkThread:workMessage
* [L7] WorkThread:postMessage
* [L7] PostThreadMessage

Worker Thread

* [L6] WorkThread:dispatchMessage – THREAD\_MSG\_DO\_EDIT
* [L6] WorkThread::GenerationStep
* [L5] EvoHistorySys::EvoCreateNextGeneration
* [L4] HistorySystemImpl::CreateAppCommand(( tEvoCmd::nextGen
* [L4] HistorySystemImpl::step2NextGeneration( tEvoCmd::nextGen
* [L3] EvoModelData::OnAppCommand( …
* [L2] EvolutionCore::Compute

### Run mode

Main Thread

* [L8] … different ways …
* [L8] EvoController::ProcessCommand( IDM\_RUN
* [L7] WorkThreadInterface:PostRunGenerations
* [L7] WorkThread:workMessage( THREAD\_MSG\_GENERATION\_RUN
* [L7] WorkThread:postMessage
* [L7] PostThreadMessage

Worker Thread

* [L6] WorkThread:dispatchMessage – THREAD\_MSG\_GENERATION\_RUN
* [L6] WorkThread:generationRun
* [L6] WorkThread::GenerationStep
* … down to EvolotionCore::Compute
* [L6] WorkThread:generationRun, loop to WorkThreadInterface:PostRunGenerations

### Goto Generation

* [L8] User action: Mouse click in history window
* [L8] HistWindow WM\_LBUTTONDOWN
* [L8] EvoHistWindow:GotoGeneration
* [L7] WorkThreadInterface::PostGotoGeneration
* [L7] WorkThreadInterface::postGotoGeneration
* [L7] WorkThread:workMessage
* [L7] WorkThread:postMessage
* [L7] PostThreadMessage

Worker Thread

* [L6] WorkThread:dispatchMessage – THREAD\_MSG\_GOTO\_GENERATION
* [L6] WorkThread::GenerationStep. Loop: PostRepeatGenerationStep
* [L5] EvoHistorySys::EvoApproachHistGen
* [L4] HistorySystemImpl::ApproachHistGen
* [L4] HistorySystemImpl::step2NextGeneration
* [L3] EvoModelData::OnAppCommand( …
* [L2] EvolutionCore::Compute

### Stop command in run mode

### Reset