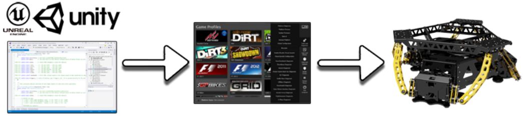
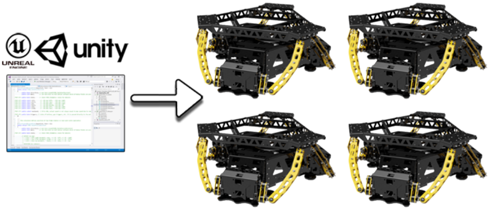
SDK

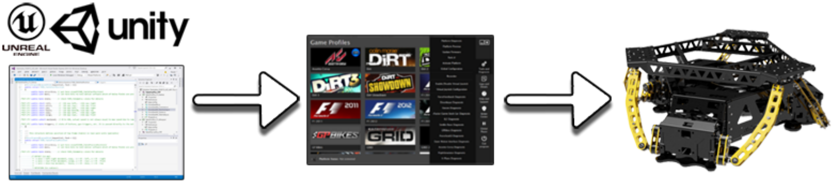
ForceSeatMI



ForceSeatDI



ForceSeatMI - Introduction



- Recommended for vehicle physics simulation
- Hardware independent - different script for different motion platforms
- Telemetry to motion transformation is done outside of the SIM
- All diagnostic and processing features of ForceSeatPM are available

ForceSeatMI - Features

- Supports forces simulation (telemetry), fast top frame positioning and precise top frame positioning
- Supports Inverse Kinematics for 3DoF and 6DoF
- Supports multiple platforms in clone mode over USB
- Telemetry data to motors position transformation done by ForceSeatPM scripting engine
- Motion cueing can be changed at runtime without recompilation
- Constant simulation frame rate in SIM is recommended (30~50 FPS)
- Feedback about current motors position is available at 20ms intervals

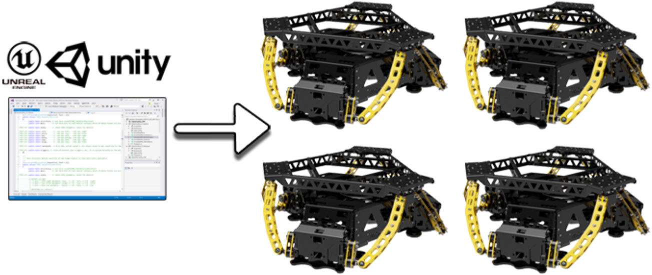
ForceSeatMI - Operation modes - Comparison

Mode	Description	Applications
Telemetry data	SIM provides g-forces and accelerations and ForceSeatPM transforms from forces to top frame movements	Vehicle physics simulations
Fast table position	SIM sends top frame position in abstract units	Positioning applications
Precise table position	SIM sends top frame position in real world units (Inverse Kinematics)	Equipment testing

ForceSeatMI - Operation modes - Features

Mode	Features
Telemetry data	Motion cuening can be changed at runtime without source code recompilation. It is easy to introduce support for different motion platforms by just changing a motion profile
Fast table position	Motion platform will always move (to the closest possible position) even when transformation is outside working range
Precise table position	Working in real world units. SIM has to make sure that requested transformation is with-in working range, otherwise top frame will not move

ForceSeatDI - Introduction



- Recommended for applications that require complex but fully synchronized movements of multiple motion platforms
- Motion cueing implemented in SIM
- No need for ForceSeatPM or other external processes

ForceSeatDI - Features

- Supports fast top frame positioning and precise top frame positioning
- Supports Inverse Kinematics for 3DoF and 6DoF
- Better control of the hardware
- Support multiple platforms over USB and over Ethernet
- Different positioning request can be send to each motion platform
- Constant simulation frame rate in SIM is recommended (30~50 FPS)
- Feedback about current motors position is being refreshed up to 100 times/sec

ForceSeatDI - Operation modes - Comparison

Mode	Description	Applications
Fast table position	SIM sends top frame position in abstract units	Positioning applications
Precise table position	SIM sends top frame position in real world units (Inverse Kinematics)	Equipment testing

ForceSeatDI - Operation modes - Features

Mode	Features
Fast table position	Motion platform will always move (to the closest possible position) even when transformation is outside working range
Precise table position	Working in real world units. SIM has to make sure that requested transformation is with-in working range, otherwise top frame will not move

ForceSeatMI vs ForceSeatDI

	ForceSeatMI	ForceSeatDI
C/C++/C#/Unity	■	■
Linux	□	■
Multiple platforms from one PC	USB (mirror)	USB, Ethernet
Error handling & diagnostic	By ForceSeatPM	By the SIM
Requires ForceSeatPM	■	□
Telemetry mode & scripting	■	□
Fast and precise positioning	■	■
Profile selection by the user	■	□
Inverse kinematics 3DoF/6DoF	■	■

SDK - Example

- Both ForceSeatMI and ForceSeatDI come as set of .h files and libraries
- Both have similar, simple in use API
- Both are delivered with C/C++ and C# examples
- Following example is for ForceSeatDI

SDK - Example - ForceSeatDI

```
// Create API object
ForceSeatDI api = new ForceSeatDI();

// Connect via Ethernet or USB
api.ConnectToNetworkDevice("10.1.1.75");
api.TestConnection(ref isConnected);

... // Operation is performed in regular intervals

// When the SIM exists, it can park the motion
// platform and dispose the API
api.Park();
api.Dispose();
api = null;
```

SDK - Example - ForceSeatDI

```
// Create a structure that will be used to
// send required position to the motion platform
var pos = new FSDI_TopTablePositionPhysical();

// Configure what information will be provided
// to the motion platform
pos.mask = FSDI_BIT.PAUSE | FSDI_BIT.POSITION |
    FSDI_BIT.MAX_SPEED;

// Initialize other fields
pos.structSize = (byte)Marshal.SizeOf(pos);
pos.maxSpeed = 65535; // maximum allowed speed
```

SDK - Example - ForceSeatDI

```
public struct FSDI_TopTablePositionPhysical
{
  public byte structSize;
  public uint mask;                // BIT field

/*BIT:1*/ public byte pause;

/*BIT:2*/ public float roll;       // in radians
/*BIT:2*/ public float pitch;      // in radians
/*BIT:2*/ public float yaw;        // in radians
/*BIT:2*/ public float heave;      // in mm
/*BIT:2*/ public float sway;       // in mm
/*BIT:2*/ public float surge;      // in mm

/*BIT:3*/ public ushort maxSpeed;  // allowed speed
}
```

SDK - Example - ForceSeatDI

```
// Create a structure that will be used to get
// primary information from the motion platform
var platformInfo = new FSDI_PlatformInfo();
platformInfo.structSize =
   (byte)Marshal.SizeOf(platformInfo);

// Create a structure that will be used to get
// current position of actuators
var actualPositions = new
   FSDI_ActualActuatorsPositionLogical();
actualPositions.structSize =
   (byte)Marshal.SizeOf(actualPositions);
```

SDK - Example - ForceSeatDI

```
public struct FSDI_PlatformInfo
{
  ...
  public byte state;
  public byte isThermalProtectionActivated;
  public byte coolingSystemMalfunction;
  public byte moduleStatus[6];
  public byte serialNumber[12];
}
public struct FSDI_ActualActuatorsPositionLogical
{
  ...
  public ushort actualMotorPosition[6];
  public int    actualMotorSpeed[6];
  public ushort requiredMotorPosition[6];
  public ushort maxAllowedMotorSpeed[6];
}
```

SDK - Example - ForceSeatDI

```
// No pause, 10 degree of roll and 50 mm of heave
pos.pause = 0; // No pause
pos.roll = Deg2Rad(10);
pos.heave = 50; // mm

// Send data to the motion platform
api.SendTopTablePosPhy(ref pos);

// Get feedback from the motion platform
api.GetPlatformInfo(ref platformInfos);
api.GetActuatorsPosLog(ref actualPositions);
```