PROJECT CHRONO

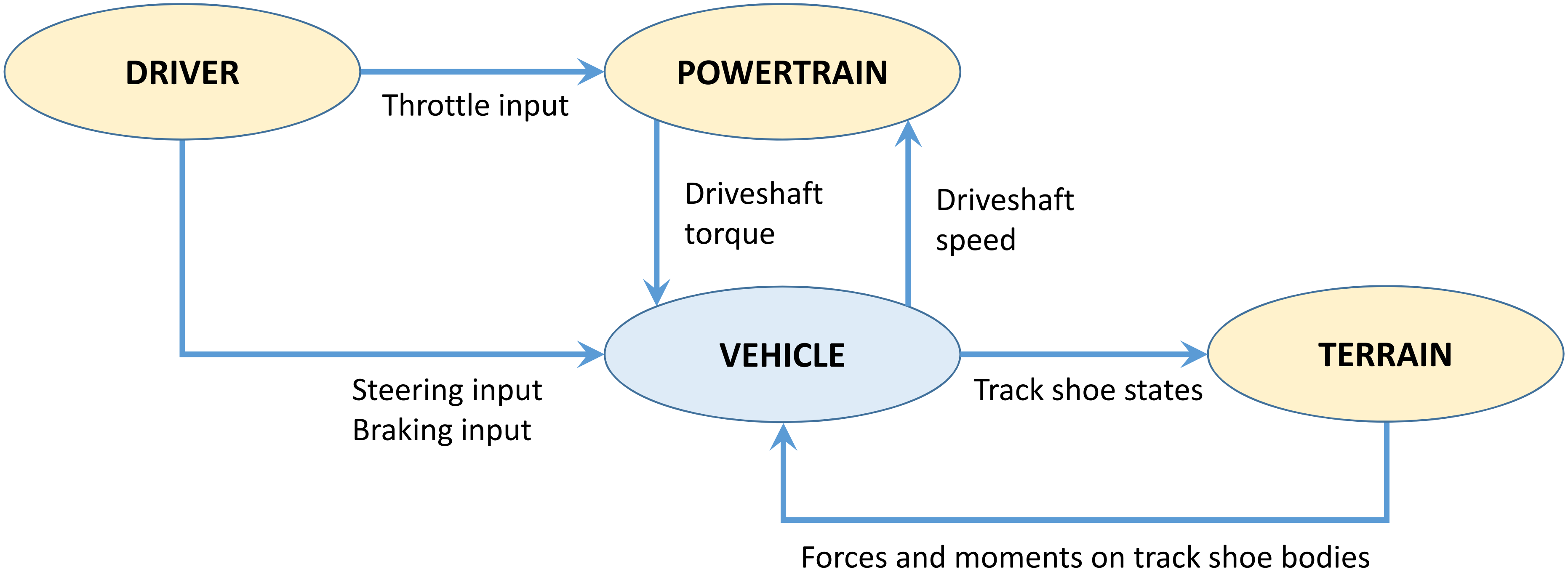
Chrono::Vehicle Tutorial

Tracked vehicle system

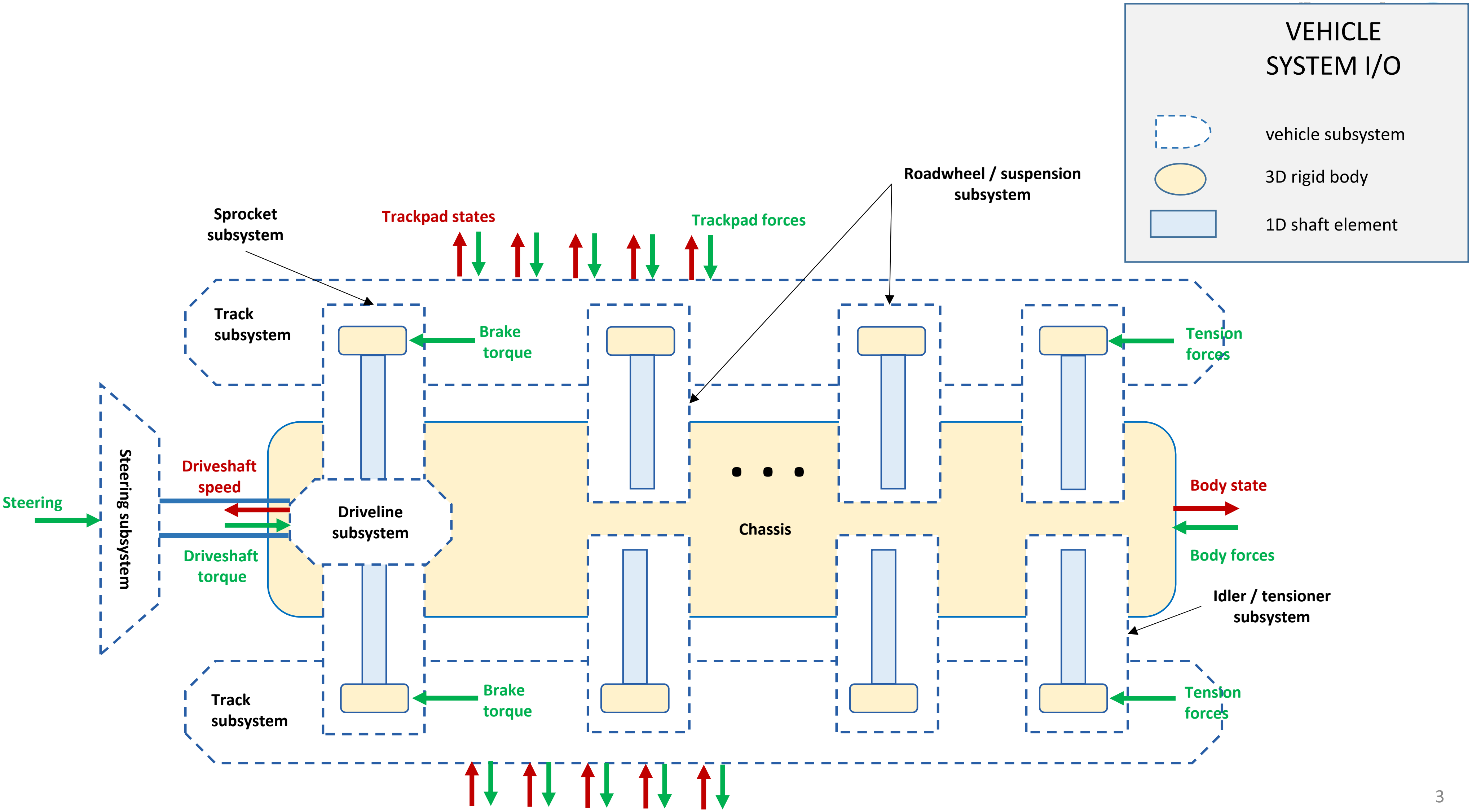
WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

UNIVERSITÀ DEGLI STUDI DI PARMA

Data flow


DRIVER
POWERTRAIN
Throttle input
Driveshaft torque
Driveshaft speed
VEHICLE
TERRAIN
Steering input
Braking input
Track shoe states
Forces and moments on track shoe bodies

VEHICLE SYSTEM I/O



vehicle subsystem

3D rigid body

1D shaft element

Sprocket subsystem

Trackpad states

Trackpad forces

Roadwheel / suspension subsystem

Track subsystem

Brake torque

Tension forces

Steering subsystem

Steering

Driveshaft speed

Driveshaft torque

Driveline subsystem

Chassis

Body state

Body forces

Idler / tensioner subsystem

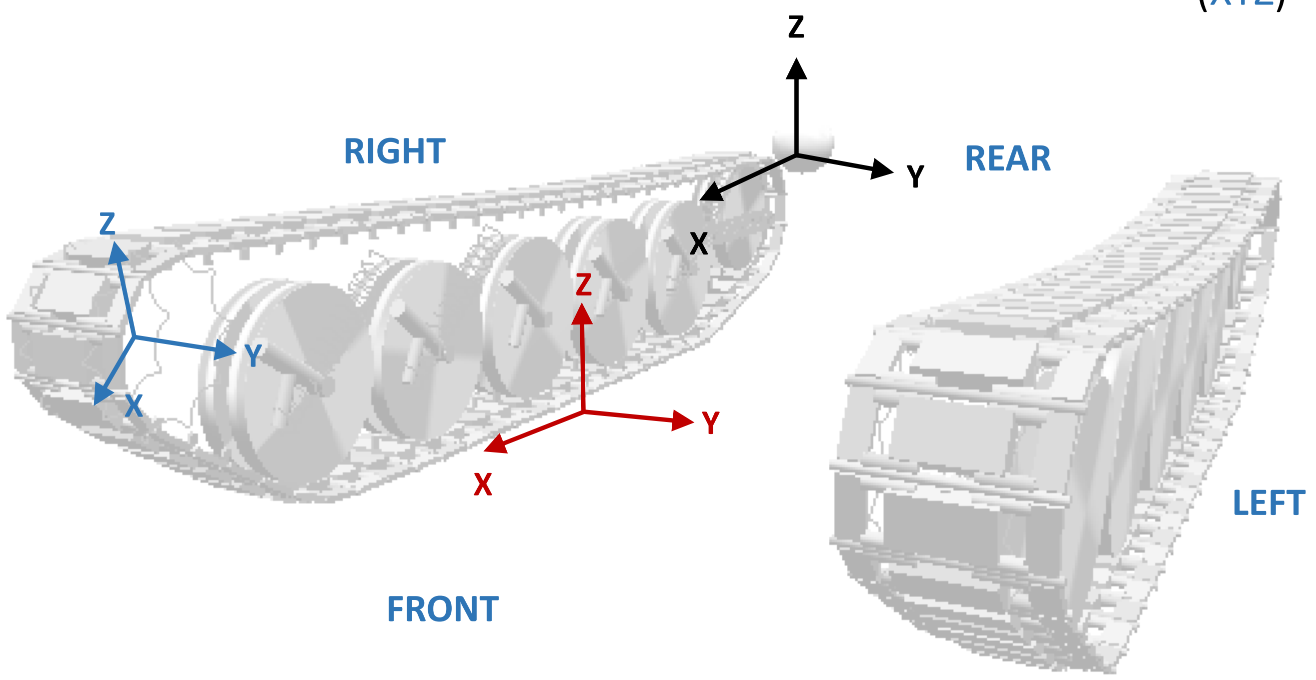
Track subsystem

Brake torque

Tension forces

Vehicle ISO reference frames

(XYZ) – vehicle (chassis) reference frame
(XYZ) – chassis COM reference frame
(XYZ) – right sprocket reference frame



ChTrackedVehicle base class

- A ChTrackedVehicle is a Chrono ChVehicle:

```
/// Base class for chrono tracked vehicle systems.
/// This class provides the interface between the vehicle system and other
/// systems (terrain, driver, etc.)
class CH_VEHICLE_API ChWheeledVehicle : public ChVehicle
```

- A ChTrackedVehicle has:

```
 std::shared_ptr<ChTrackAssembly> m_tracks[2];   ///< handles to the track assemblies (left/right)
 std::shared_ptr<ChTrackDriveline> m_driveline;  ///< handle to the driveline subsystem

 ChTrackContactManager* m_contacts;              ///< manager for internal contacts
```

ChTrackedVehicle base class accessors

- Deferring to its constituent subsystems as needed, a ChTrackedVehicle provides accessors for:
  - Vehicle subsystems
  - States of the vehicle’s track shoe bodies
  - Inherited accessors from ChVehicle

- A ChTrackedVehicle intermediates communication between other systems (e.g., powertrain, driver, etc.) and constituent subsystems (e.g., sprockets, driveline, brakes, etc.)

ChTrackedVehicle base class virtual functions

- Synchronize the vehicle at a communication time with data from other systems

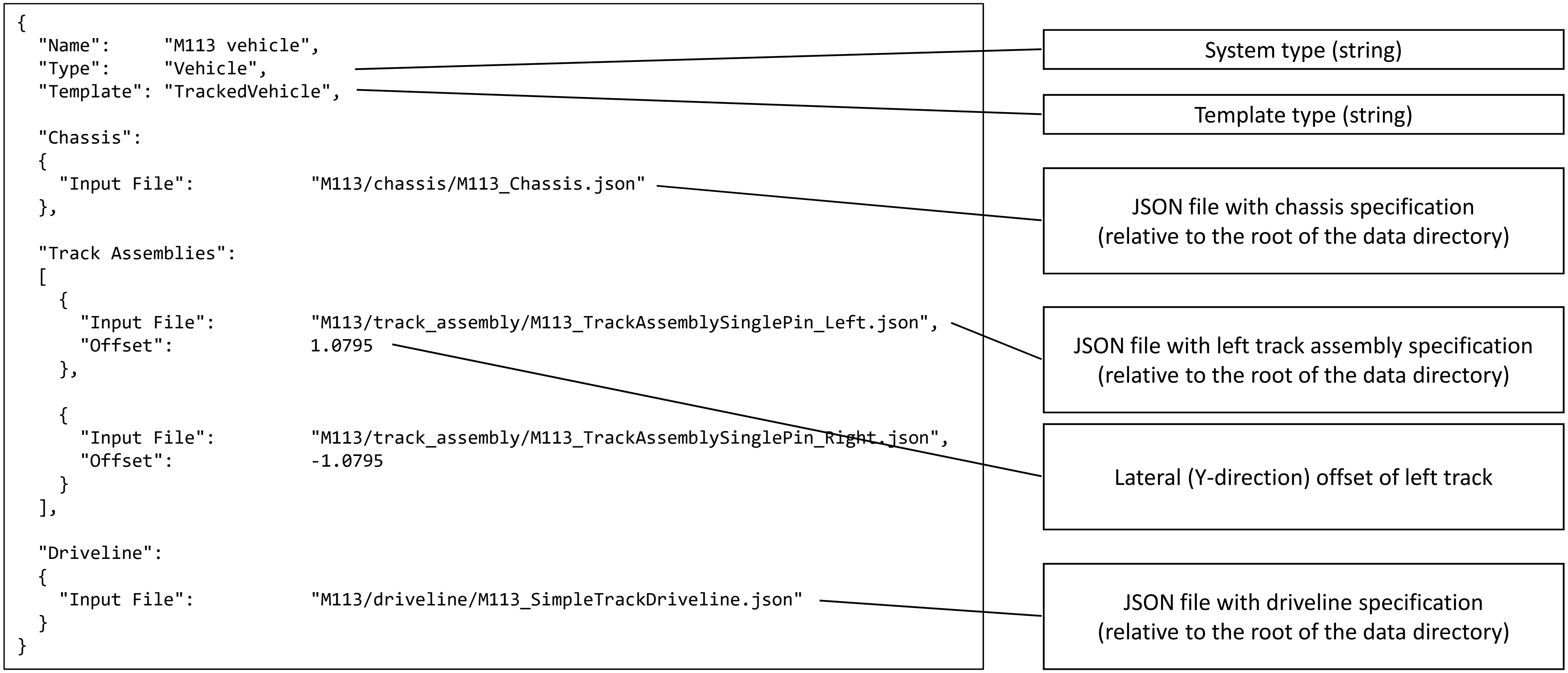
```
    /// Update the state of this vehicle at the current time.
    /// The vehicle system is provided the current driver inputs (throttle between
    /// 0 and 1, steering between -1 and +1, braking between 0 and 1), the torque
    /// from the powertrain, and tire forces (expressed in the global reference
    /// frame).
    void Synchronize(double time,                              ///< [in] current time
                     double steering,                          ///< [in] current steering input [-1,+1]
                     double braking,                           ///< [in] current braking input [0,1]
                     double powertrain_torque,                 ///< [in] input torque from powertrain
                     const TrackShoeForces& shoe_forces_left,  ///< [in] vector of track shoe forces (left side)
                     const TrackShoeForces& shoe_forces_right  ///< [in] vector of track shoe forces (left side)
                     );
```

Data exchange structures

- TrackShoeForce structure – encapsulates external forces applied to a track shoe body
  - Force vector and application point (expressed in the global reference frame)
  - Moment vector (expressed in the global reference frame)
- A track shoe force structure can be specified for any (or all) track shoes (e.g., to model track-terrain contact forces)
  - The force and moment are applied to the track shoe body as external forces

```
/// Structure to communicate a set of generalized track shoe forces.
struct TrackShoeForce {
    ChVector<> force;   ///< force vector, epxressed in the global frame
    ChVector<> point;   ///< global location of the force application point
    ChVector<> moment;  ///< moment vector, expressed in the global frame
};
```

JSON specification file for a tracked vehicle



```
{
  "Name":     "M113 vehicle",
  "Type":     "Vehicle",
  "Template": "TrackedVehicle",

  "Chassis":
  {
    "Input File":           "M113/chassis/M113_Chassis.json"
  },

  "Track Assemblies":
  [
    {
      "Input File":         "M113/track_assembly/M113_TrackAssemblySinglePin_Left.json",
      "Offset":             1.0795
    },

    {
      "Input File":         "M113/track_assembly/M113_TrackAssemblySinglePin_Right.json",
      "Offset":             -1.0795
    }
  ],

  "Driveline":
  {
    "Input File":           "M113/driveline/M113_SimpleTrackDriveline.json"
  }
}
```

System type (string)

Template type (string)

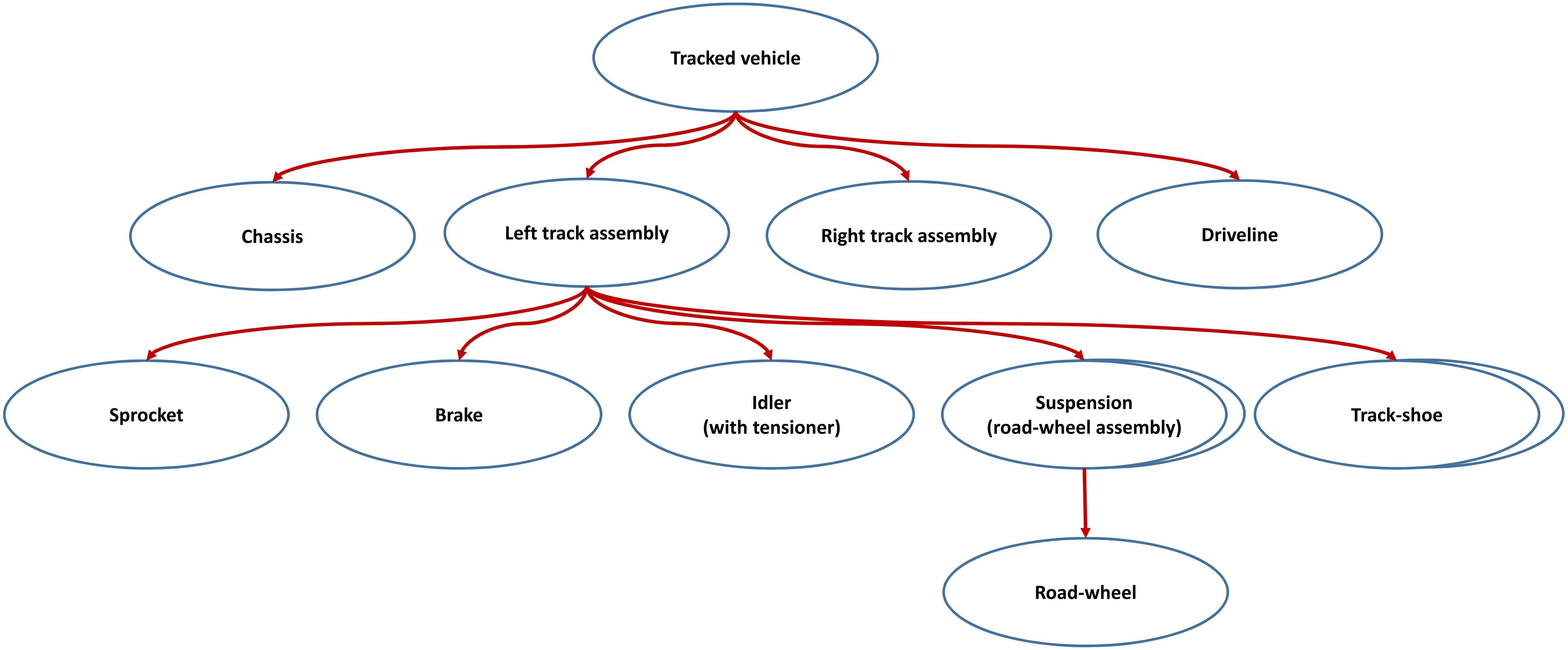
JSON file with chassis specification
(relative to the root of the data directory)

JSON file with left track assembly specification
(relative to the root of the data directory)

Lateral (Y-direction) offset of left track

JSON file with driveline specification
(relative to the root of the data directory)

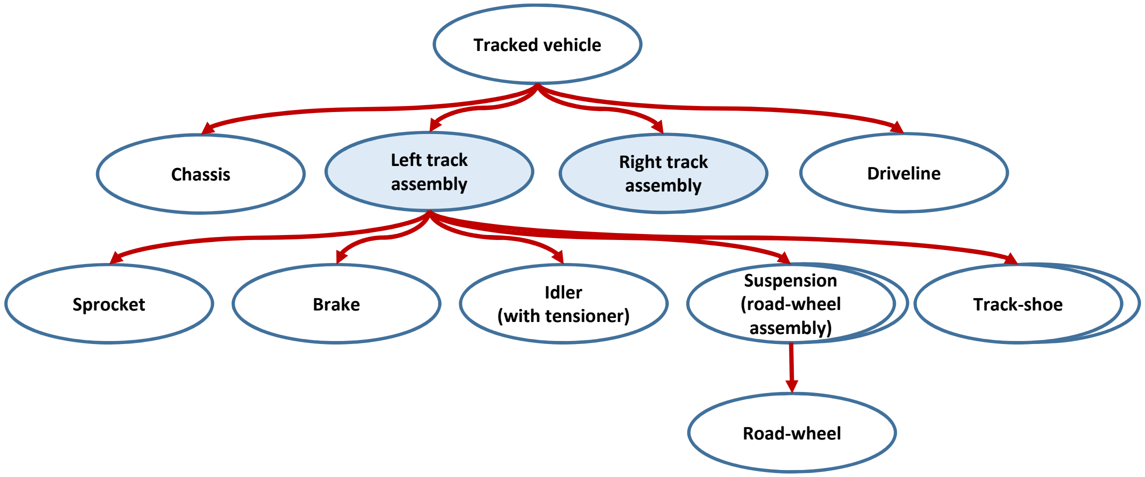
Tracked vehicle subsystem hierarchy



Subsystem dependencies

- Sprocket ↔ track-shoe
  - Sprocket type and track-shoe type must match:
    - “single-pin”
    - “double-pin”
  - Contact between sprocket and track shoes is implemented through a custom callback which assumes consistency

- Sprocket/Idler/road-wheel ↔ track-shoe
  - Wheel type and track-shoe type must match:
    - “single-wheel” and “lateral guiding pin”
    - “double-wheel” and “central guiding pin”
  - Note: track shoes with lateral guiding pins currently not implemented



Track Assembly Subsystem

ChTrackAssembly base class

- ChTrackAssembly is a composite class, used to manage all subsystems comprising a (left or right) track assembly:
  - A sprocket and brake
  - An idler assembly (idler wheel + tensioner mechanism)
  - A set of suspensions (each containing a road-wheel)
  - A set of track shoes
- Derived classes ensure consistency between subsystem types
- ChTrackAssembly provides the algorithm for assembling the track shoes around the wheels (sprocket, idler, road-wheels)

```
/// Definition of a track assembly.
/// A track assembly consists of a sprocket, an idler (with tensioner mechanism),
/// a set of suspensions (road-wheel assemblies), and a collection of track shoes.
class CH_VEHICLE_API ChTrackAssembly : public ChPart
```

ChTrackAssembly class members

- A ChTrackAssembly has:

```
VehicleSide m_side;                     ///< assembly on left/right vehicle side
std::shared_ptr<ChIdler> m_idler;       ///< idler (and tensioner) subsystem
std::shared_ptr<ChTrackBrake> m_brake;  ///< sprocket brake
ChRoadWheelAssemblyList m_suspensions;  ///< road-wheel assemblies
```

- Derived classes (track assembly templates) manage the sprocket and track shoes of appropriate types

ChTrackAssembly base class accessors

- A ChTrackAssembly provides access to:
  - Its constituent subsystems (sprocket, brake, idler, suspensions, individual track shoes)
    - Sprocket and track shoe access provided through pure virtual methods
  - Relative positions of its constituent subsystems
    - The ISO track assembly reference frame is assumed to have origin at the center of the sprocket
  - Complete state of a track shoe subsystem (through its index in the vector of track shoes in the assembly)
  - Cumulative mass of the track assembly

ChTrackAssembly base class methods

- A ChTrackAssembly provides methods to:

```
    /// Initialize this track assembly subsystem.
    /// The subsystem is initialized by attaching it to the specified chassis body
    /// at the specified location (with respect to and expressed in the reference
    /// frame of the chassis). It is assumed that the track assembly reference frame
    /// is always aligned with the chassis reference frame.
    void Initialize(std::shared_ptr<ChBodyAuxRef> chassis,  ///< [in] handle to the chassis body
                    const ChVector<>& location              ///< [in] location relative to the chassis frame
                    );

    /// Update the state of this track assembly at the current time.
    void Synchronize(double time,                        ///< [in] current time
                     double braking,                     ///< [in] braking driver input
                     const TrackShoeForces& shoe_forces  ///< [in] vector of tire force structures
                     );
```

ChTrackAssembly base class virtual methods

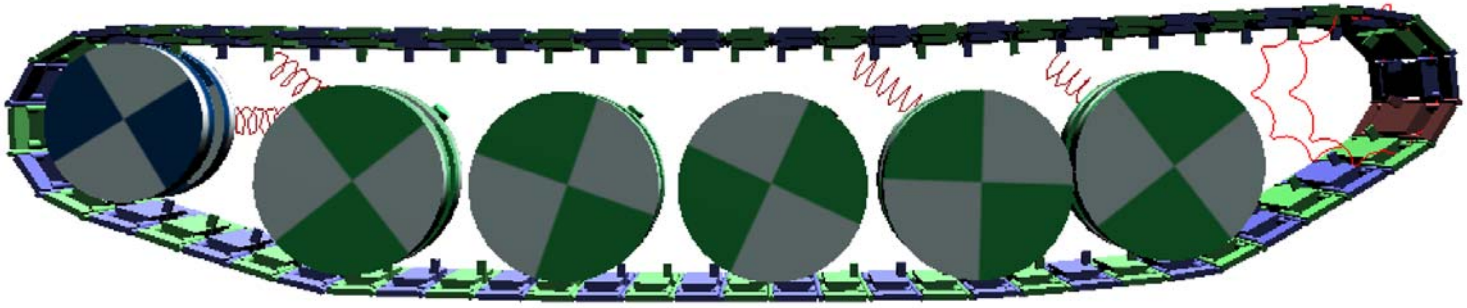
- A derived class must provide a method for assembling track shoes around the assembly’s wheels

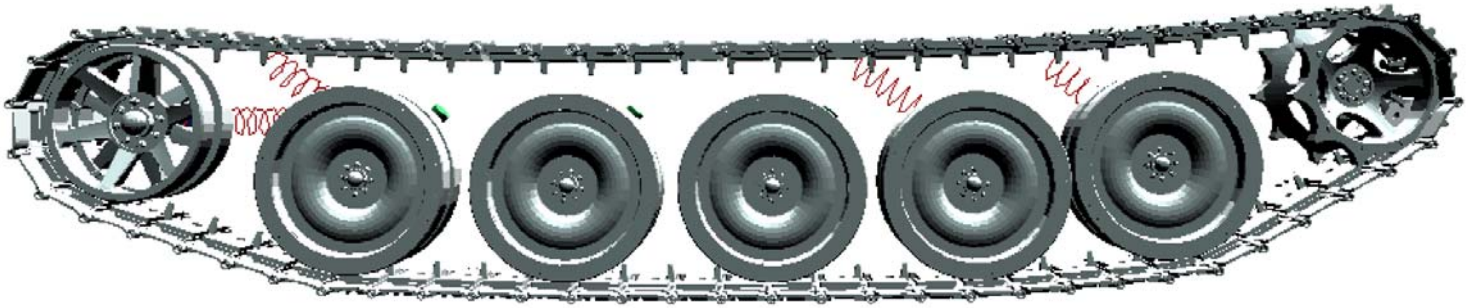
```
    /// Assemble track shoes over wheels.
    /// Return true if the track shoes were initialized in a counter clockwise
    /// direction and false otherwise.
    virtual bool Assemble(std::shared_ptr<ChBodyAuxRef> chassis) = 0;
```

- Track shoes are positioned from below the sprocket, clockwise or counter-clockwise, depending on whether the assembly has a front or rear sprocket
- Note that this process is relatively fragile
    - May require adjustments to initial idler position

Track Assembly Templates

Single-pin





JSON specification for single-pin track assembly (1/2)

```
{
  "Name":                       "M113 SinglePin TrackAssembly Left",
  "Type":                       "TrackAssembly",
  "Template":                   "TrackAssemblySinglePin",

  "Sprocket":
  {
      "Input File":             "M113/sprocket/M113_SprocketSinglePin_Left.json",
      "Location":               [0, 0, 0]
  },

  "Brake":
  {
      "Input File":             "M113/brake/M113_TrackBrakeSimple.json"
  },

  "Idler":
  {
      "Input File":             "M113/idler/M113_DoubleIdler_Left.json",
      "Location":               [-3.83, 0, -0.12]
  },

 "Track Shoes":
  {
      "Input File":             "M113/track_shoe/M113_TrackShoeSinglePin.json",
      "Number Shoes":           63
  },
```

System type (string)

Template type (string)

Location of sprocket subsystem
with respect to assembly reference frame

Location of idler subsystem
with respect to assembly reference frame

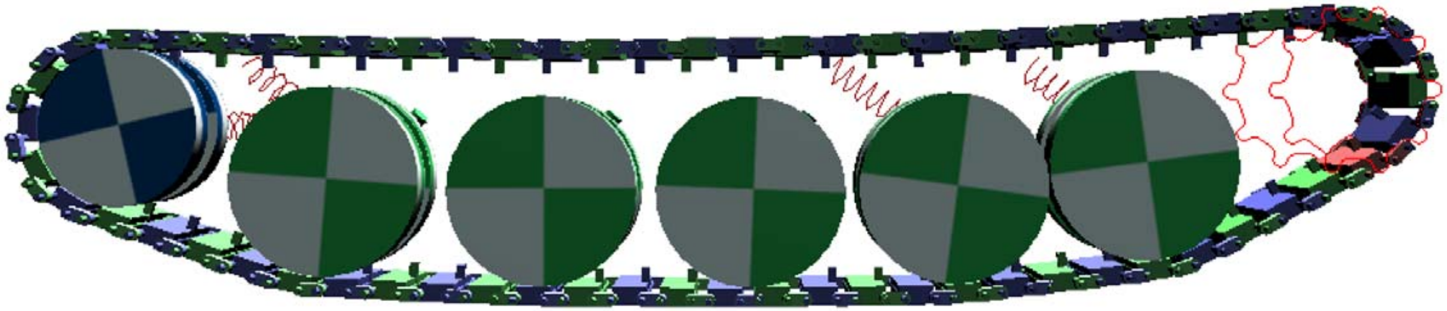
JSON specification for single-pin track assembly (2/2)

```
  "Suspension Subsystems":
  [
    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          true,
        "Location":           [-0.655, 0, -0.215]
    },

    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          true,
        "Location":           [-1.322, 0, -0.215]
    },

    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          false,
        "Location":           [-1.989, 0, -0.215]
    },

    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          false,
        "Location":           [-2.656, 0, -0.215]
    },

    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          true,
        "Location":           [-3.322, 0, -0.215]
    }
  ]
}
```

Location of first (front) suspension subsystem with respect to assembly reference frame

Track Assembly Templates

Double-pin

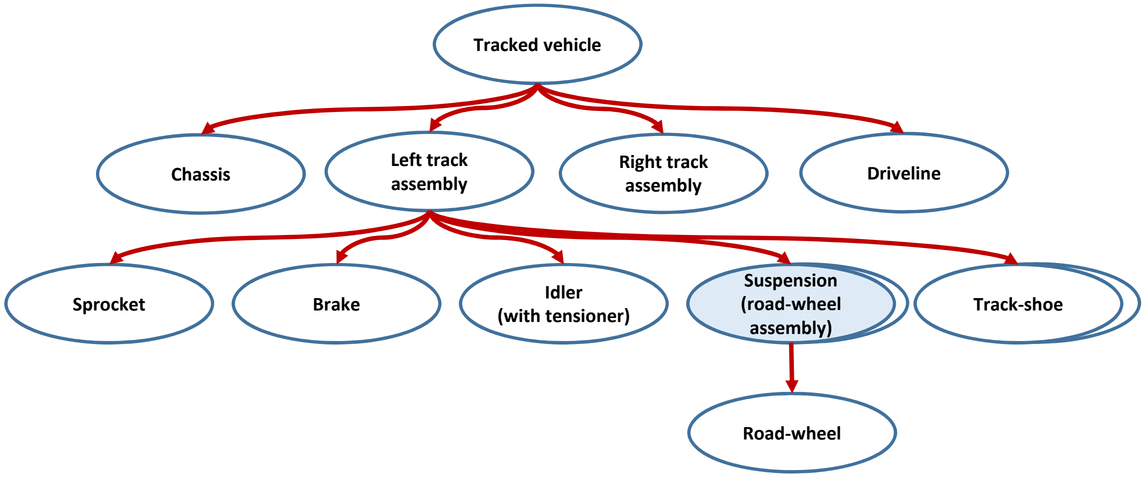


JSON specification for double-pin track assembly (1/2)

```
{
  "Name":                        "M113 DoublePin TrackAssembly Left",
  "Type":                        "TrackAssembly",
  "Template":                    "TrackAssemblyDoublePin",

  "Sprocket":
  {
      "Input File":              "M113/sprocket/M113_SprocketDoublePin_Left.json",
      "Location":                [0, 0, 0]
  },

  "Brake":
  {
      "Input File":              "M113/brake/M113_TrackBrakeSimple.json"
  },

  "Idler":
  {
      "Input File":              "M113/idler/M113_DoubleIdler_Left.json",
      "Location":                [-3.83, 0, -0.12]
  },

 "Track Shoes":
  {
      "Input File":              "M113/track_shoe/M113_TrackShoeDoublePin.json",
      "Number Shoes":            63
  },
```

JSON specification for double-pin track assembly (2/2)

```
  "Suspension Subsystems":
  [
    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          true,
        "Location":           [-0.655, 0, -0.215]
    },

    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          true,
        "Location":           [-1.322, 0, -0.215]
    },

    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          false,
        "Location":           [-1.989, 0, -0.215]
    },

    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          false,
        "Location":           [-2.656, 0, -0.215]
    },

    {
        "Input File":         "M113/suspension/M113_LinearDamperSuspension_Left.json",
        "Has Shock":          true,
        "Location":           [-3.322, 0, -0.215]
    }
  ]
}
```



Suspension Subsystem

ChRoadWheelAssembly base class

- Base class for track suspension subsystems
- Provides access to the underlying road-wheel subsystem and its components (body and revolute joint)

```
/// Base class for tracked vehicle suspension (road-wheel assembly) subsystem.
class CH_VEHICLE_API ChRoadWheelAssembly : public ChPart
```

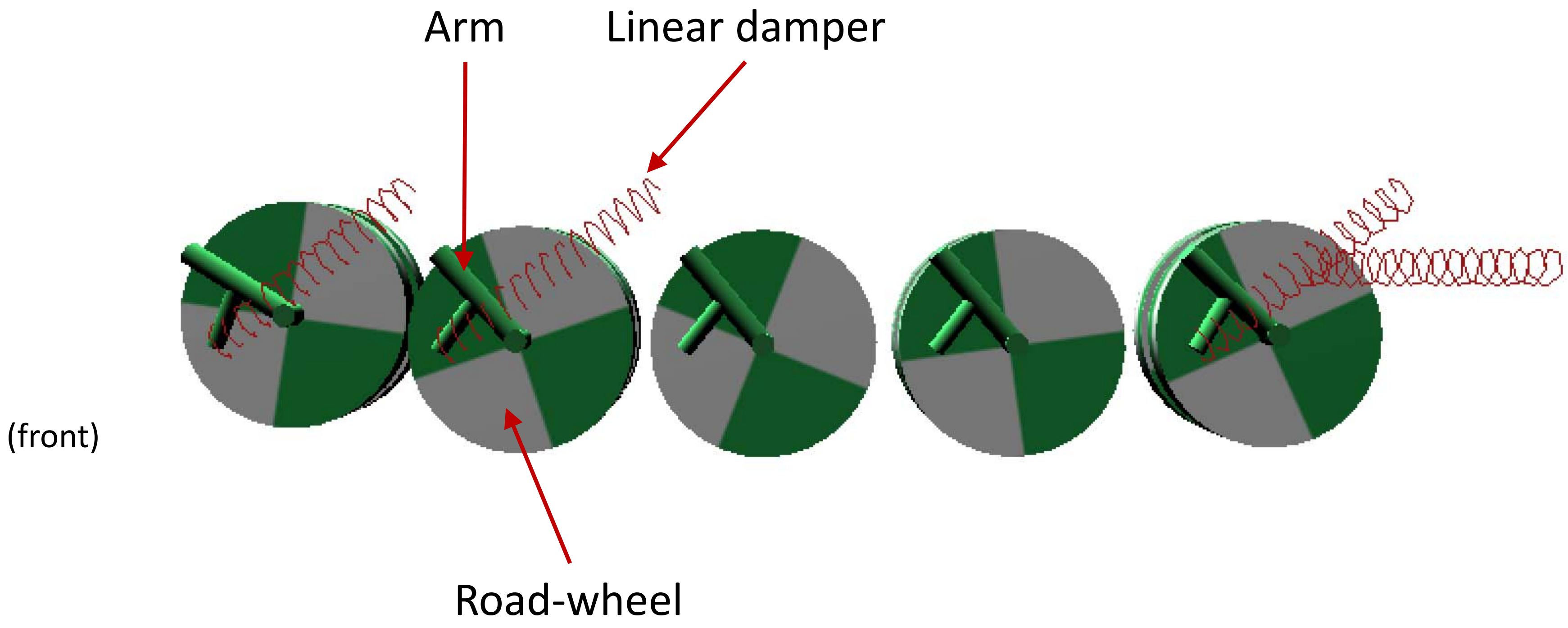
ChRoadWheelAssembly class members

- A ChRoadWheelAssembly has:

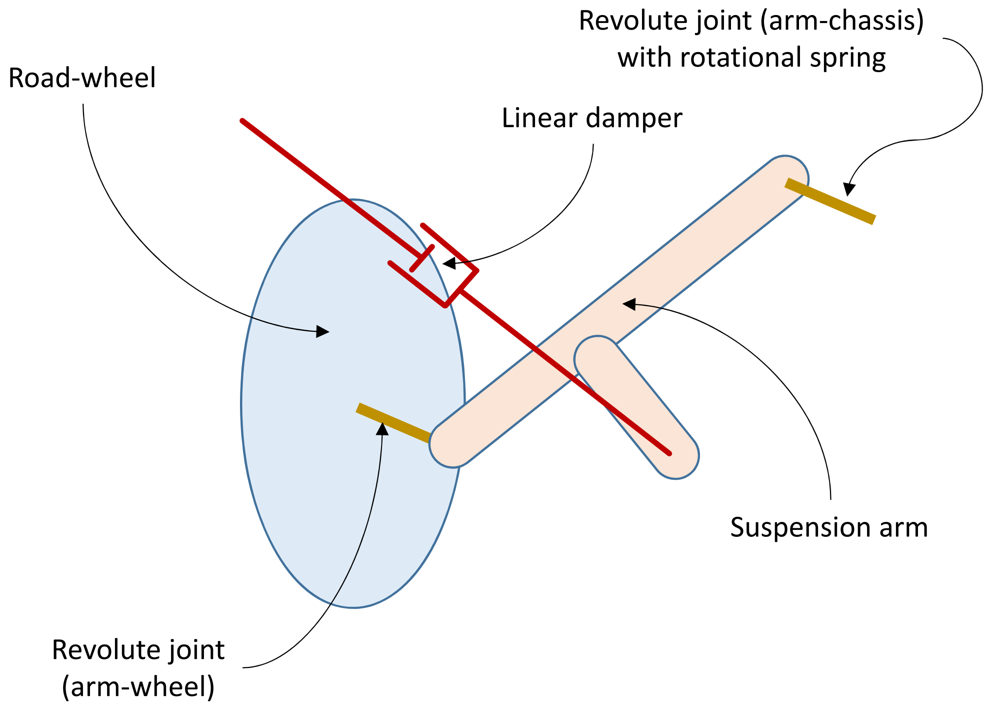
```
GuidePinType m_type;                         ///< type of the track shoe matching this road wheel
std::shared_ptr<ChRoadWheel> m_road_wheel;   ///< road-wheel subsystem
```

Suspension Templates

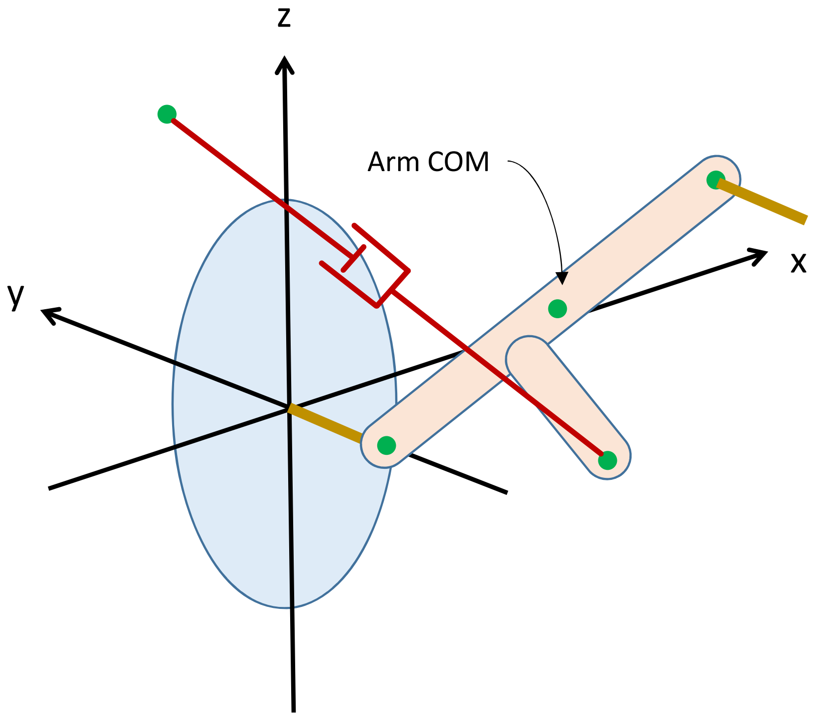
Linear-damper suspension


Arm
Linear damper
(front)
Road-wheel

Template specification



Components



Hard points

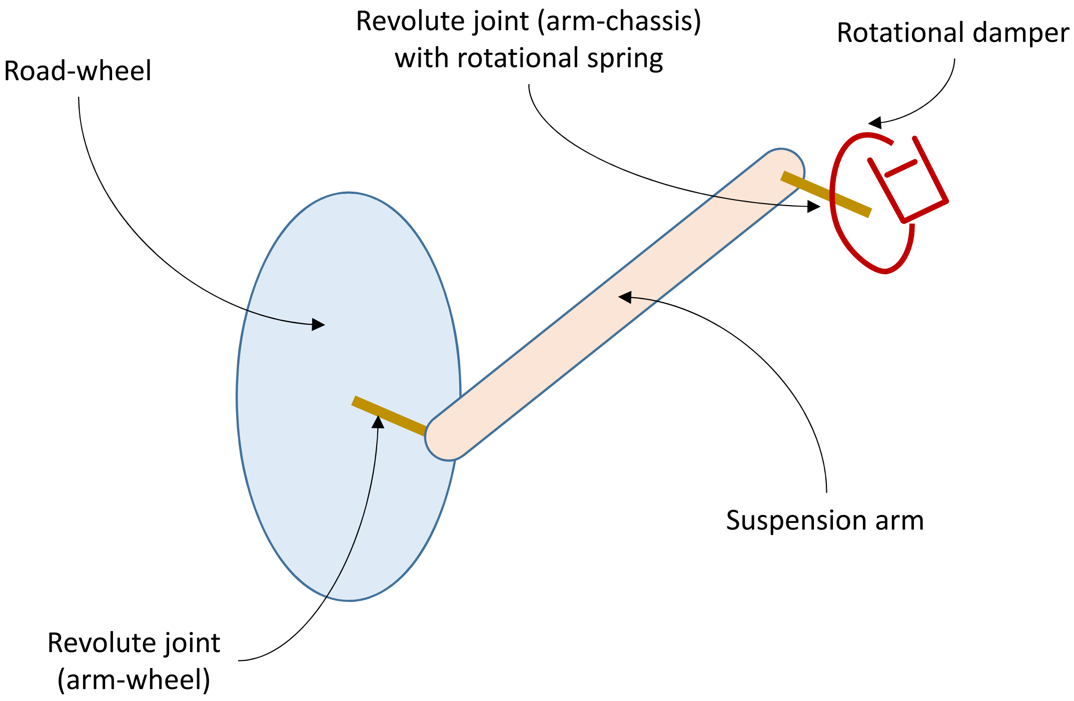
JSON specification for linear-damper suspension

```
{
  "Name":                           "M113 Linear Damper Suspension Left",
  "Type":                           "RoadWheelAssembly",
  "Template":                       "LinearDamperRWAssembly",

  "Suspension Arm":
  {
      "Mass":                       75.26,
      "COM":                        [0.144, -0.12, 0.067],
      "Inertia":                    [0.37, 0.77, 0.77],
      "Location Chassis":           [0.288, -0.12, 0.134],
      "Location Wheel":             [0, -0.12, 0],
      "Radius":                     0.03
  },

  "Torsional Spring":
  {
      "Spring Constant":            2.5e4,
      "Damping Coefficient":        5e2,
      "Preload":                    -1e4
  },

  "Damper":
  {
      "Location Chassis":           [-0.3, -0.12, 0.3],
      "Location Arm":               [0.184, -0.12, -0.106],
      "Damping Coefficient":        1e2
  },

  "Road Wheel Input File":          "M113/road_wheel/M113_DoubleRoadWheel_Left.json"
}
```

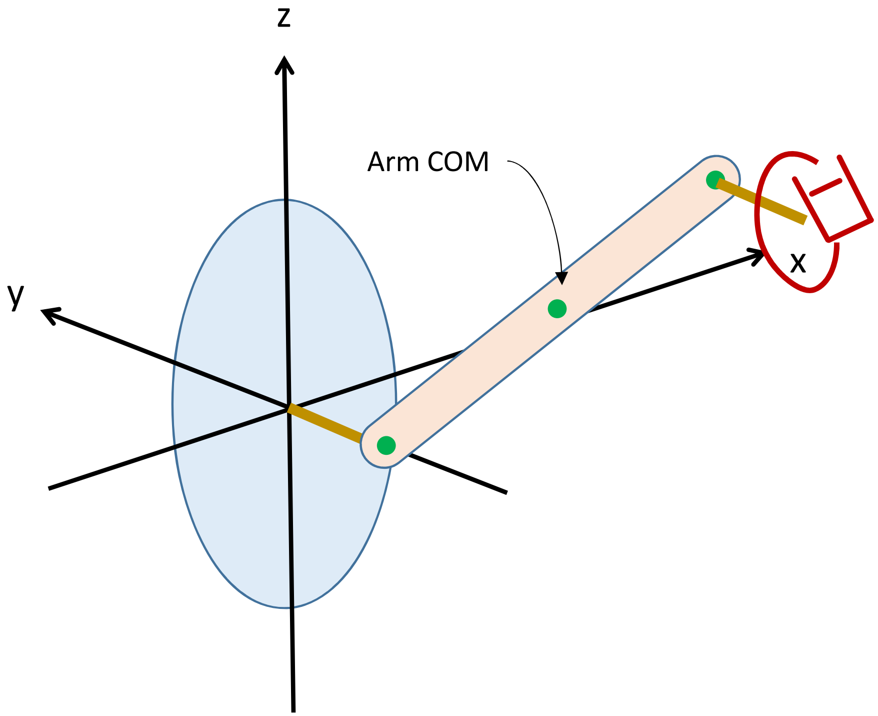
Suspension Templates

Rotational-damper suspension

Template specification



Components

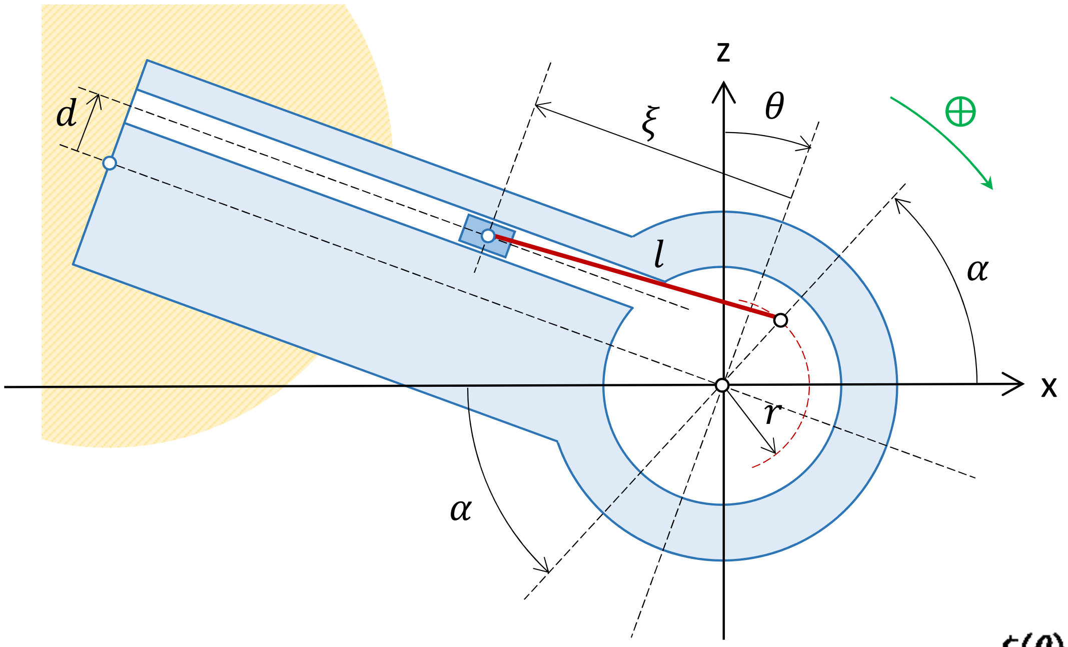


Hard points

Suspension Templates

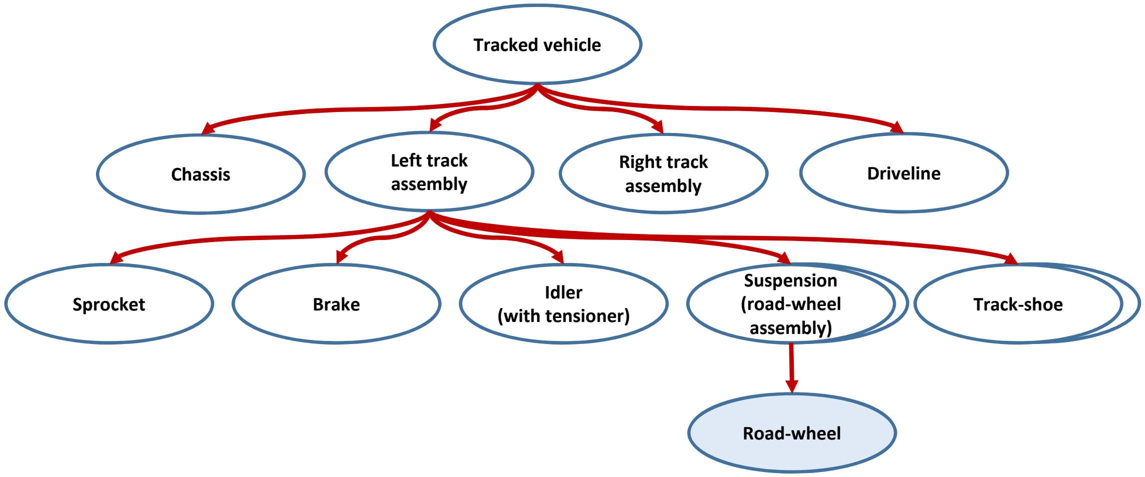
Hydropneumatic suspension

Template specification



$$
\begin{aligned}
\xi(\theta) &= \sqrt{l^2 - \left[r\sin(\theta-\alpha) - d\right]^2} - r\cos(\theta-\alpha) \\
\theta^* &= \alpha - \arctan\left(\frac{d}{l-r}\right)
\end{aligned}
$$

Note: **WIP**



Road-wheel Subsystem

ChRoadWheel base class

- A road wheel is a single rigid body with contact shape specified by a concrete subsystem template class

```
/// Base class for a road wheel subsystem.
class CH_VEHICLE_API ChRoadWheel : public ChPart
```

- Member variables

```
std::shared_ptr<ChBody> m_wheel;                  ///< handle to the road wheel body
std::shared_ptr<ChLinkLockRevolute> m_revolute;  ///< handle to wheel revolute joint

float m_friction;       ///< contact coefficient of friction
float m_restitution;    ///< contact coefficient of restitution
float m_young_modulus;  ///< contact material Young modulus
float m_poisson_ratio;  ///< contact material Poisson ratio
float m_kn;             ///< normal contact stiffness
float m_gn;             ///< normal contact damping
float m_kt;             ///< tangential contact stiffness
float m_gt;             ///< tangential contact damping
```

ChRoadWheel class members

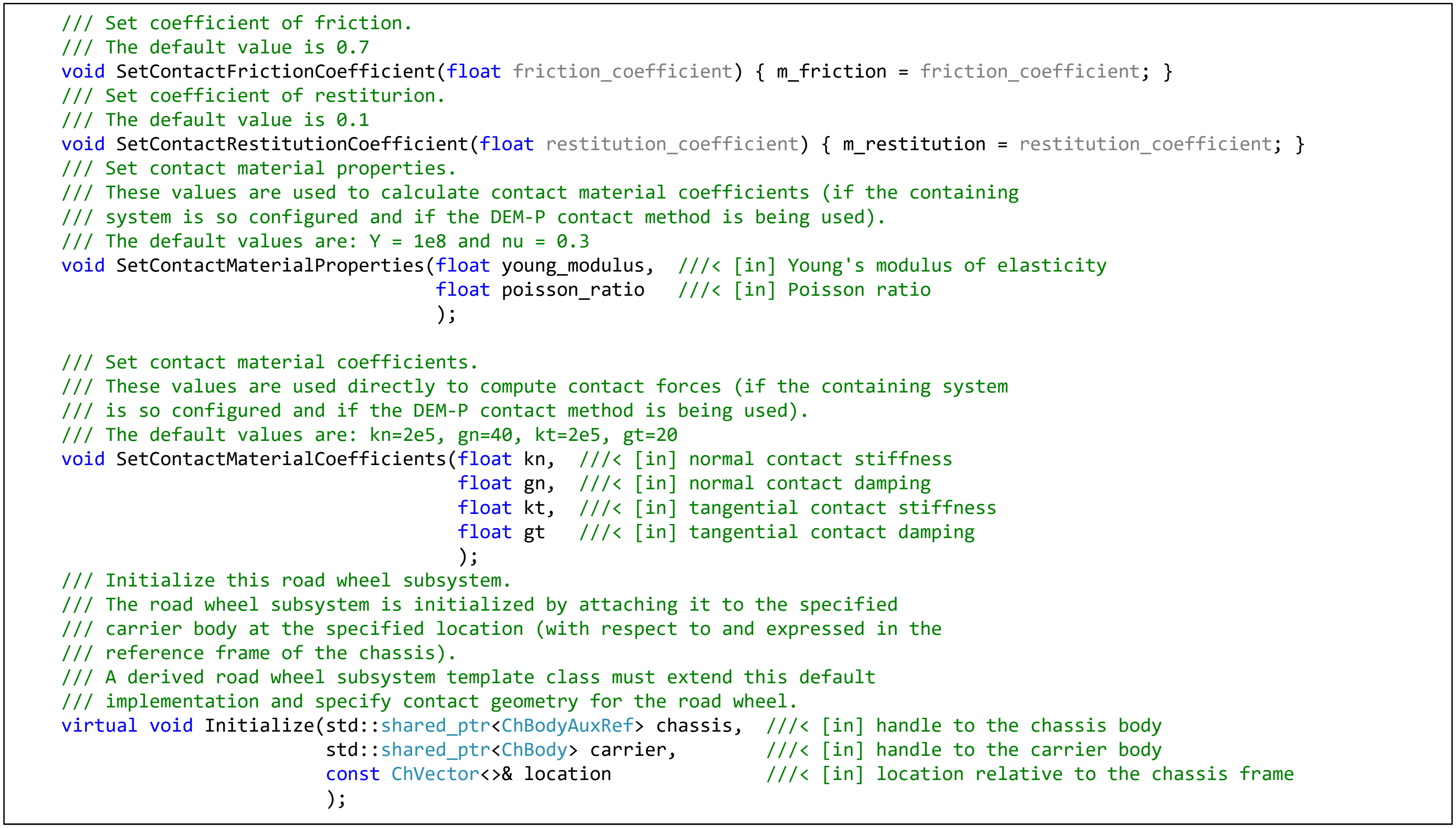
- A ChRoadWheel has:

```
std::shared_ptr<ChBody> m_wheel;                 ///< handle to the road wheel body
std::shared_ptr<ChLinkLockRevolute> m_revolute;  ///< handle to wheel revolute joint

float m_friction;       ///< contact coefficient of friction
float m_restitution;    ///< contact coefficient of restitution
float m_young_modulus;  ///< contact material Young modulus
float m_poisson_ratio;  ///< contact material Poisson ratio
float m_kn;             ///< normal contact stiffness
float m_gn;             ///< normal contact damping
float m_kt;             ///< tangential contact stiffness
float m_gt;             ///< tangential contact damping
```

ChRoadWheel base class accessor methods

```
    /// Get a handle to the road wheel body.
    std::shared_ptr<ChBody> GetWheelBody() const { return m_wheel; }

    /// Get a handle to the revolute joint.
    std::shared_ptr<ChLinkLockRevolute> GetRevolute() const { return m_revolute; }

    /// Return the mass of the road wheel body.
    virtual double GetWheelMass() const = 0;

    /// Return the moments of inertia of the road wheel body.
    virtual const ChVector<>& GetWheelInertia() = 0;

    /// Get the radius of the road wheel.
    virtual double GetWheelRadius() const = 0;

    /// Get coefficient of friction for contact material.
    float GetCoefficientFriction() const { return m_friction; }
    /// Get coefficient of restitution for contact material.
    float GetCoefficientRestitution() const { return m_restitution; }
    /// Get Young's modulus of elasticity for contact material.
    float GetYoungModulus() const { return m_young_modulus; }
    /// Get Poisson ratio for contact material.
    float GetPoissonRatio() const { return m_poisson_ratio; }
    /// Get normal stiffness coefficient for contact material.
    float GetKn() const { return m_kn; }
    /// Get tangential stiffness coefficient for contact material.
    float GetKt() const { return m_kt; }
    /// Get normal viscous damping coefficient for contact material.
    float GetGn() const { return m_gn; }
    /// Get tangential viscous damping coefficient for contact material.
    float GetGt() const { return m_gt; }
```

ChRoadWheel base class methods

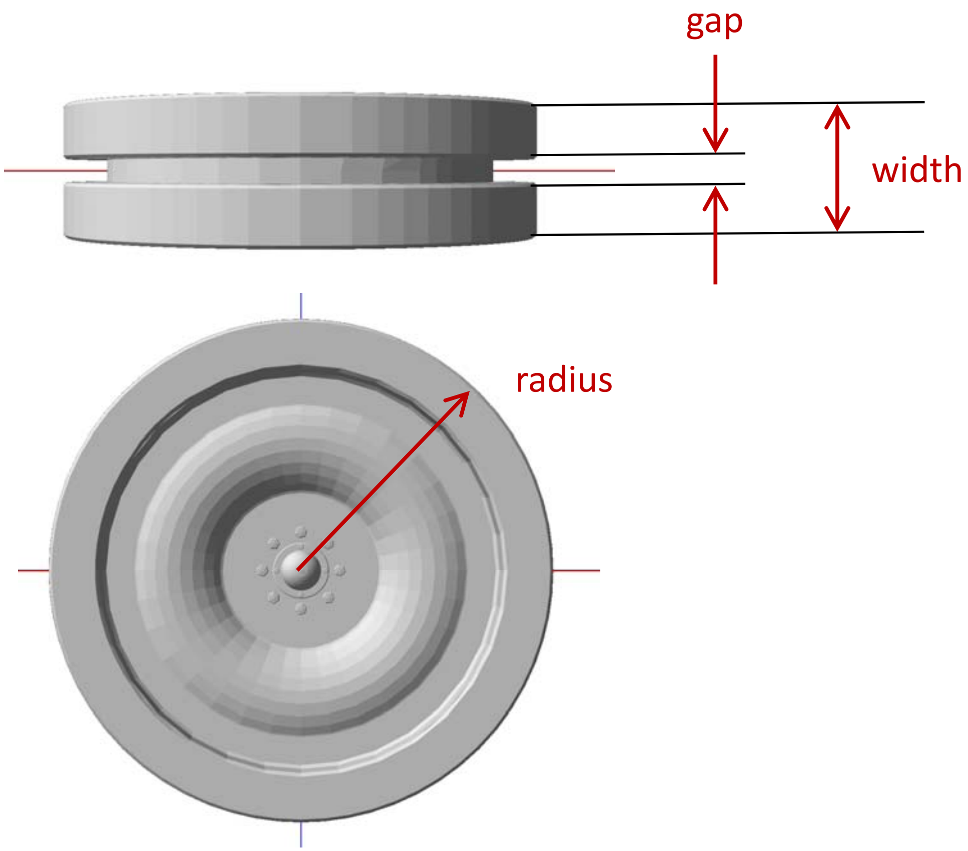


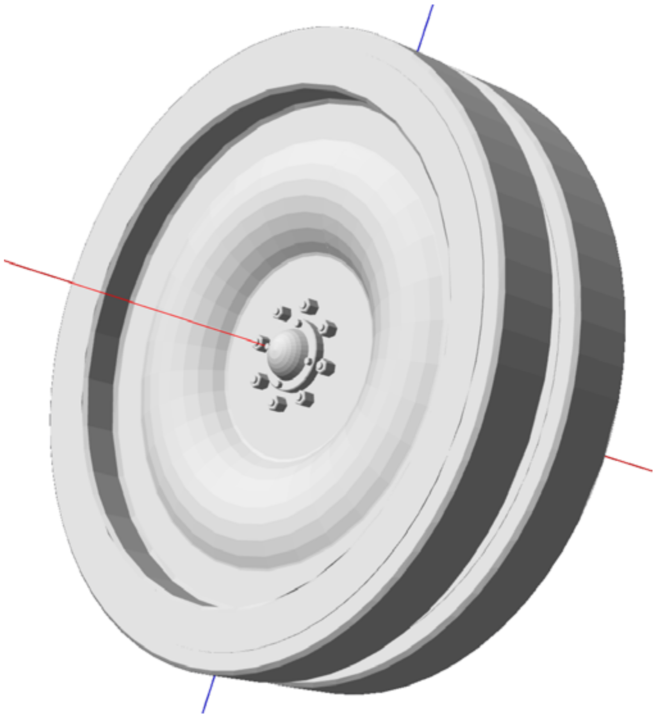
```
/// Set coefficient of friction.
/// The default value is 0.7
void SetContactFrictionCoefficient(float friction_coefficient) { m_friction = friction_coefficient; }
/// Set coefficient of restiturion.
/// The default value is 0.1
void SetContactRestitutionCoefficient(float restitution_coefficient) { m_restitution = restitution_coefficient; }
/// Set contact material properties.
/// These values are used to calculate contact material coefficients (if the containing
/// system is so configured and if the DEM-P contact method is being used).
/// The default values are: Y = 1e8 and nu = 0.3
void SetContactMaterialProperties(float young_modulus,  ///< [in] Young's modulus of elasticity
                                  float poisson_ratio   ///< [in] Poisson ratio
                                  );

/// Set contact material coefficients.
/// These values are used directly to compute contact forces (if the containing system
/// is so configured and if the DEM-P contact method is being used).
/// The default values are: kn=2e5, gn=40, kt=2e5, gt=20
void SetContactMaterialCoefficients(float kn,  ///< [in] normal contact stiffness
                                    float gn,  ///< [in] normal contact damping
                                    float kt,  ///< [in] tangential contact stiffness
                                    float gt   ///< [in] tangential contact damping
                                    );
/// Initialize this road wheel subsystem.
/// The road wheel subsystem is initialized by attaching it to the specified
/// carrier body at the specified location (with respect to and expressed in the
/// reference frame of the chassis).
/// A derived road wheel subsystem template class must extend this default
/// implementation and specify contact geometry for the road wheel.
virtual void Initialize(std::shared_ptr<ChBodyAuxRef> chassis,  ///< [in] handle to the chassis body
                        std::shared_ptr<ChBody> carrier,        ///< [in] handle to the carrier body
                        const ChVector<>& location              ///< [in] location relative to the chassis frame
                        );
```

Road-wheel Templates

Double road-wheel

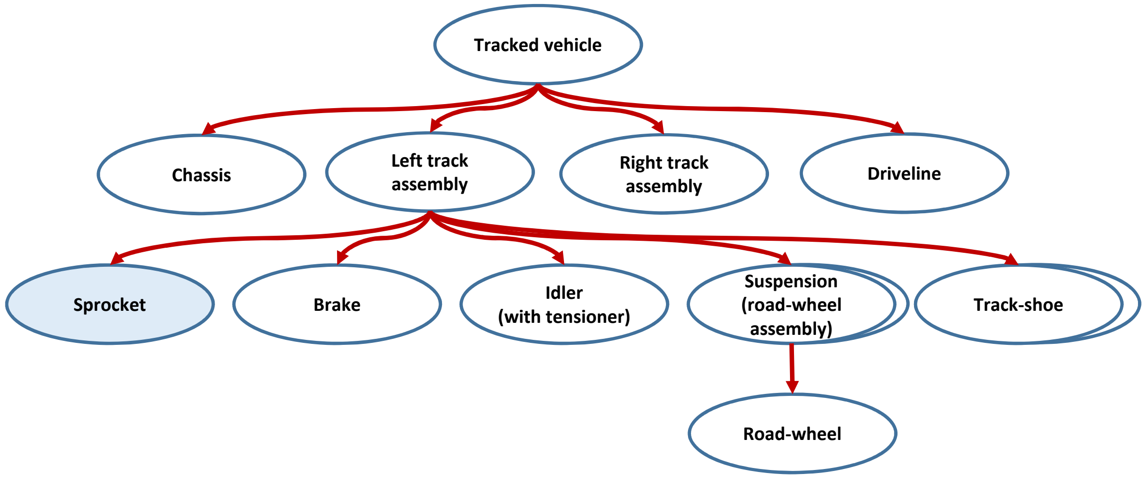
ChDoubleRoadWheel geometry





JSON specification for double road-wheel

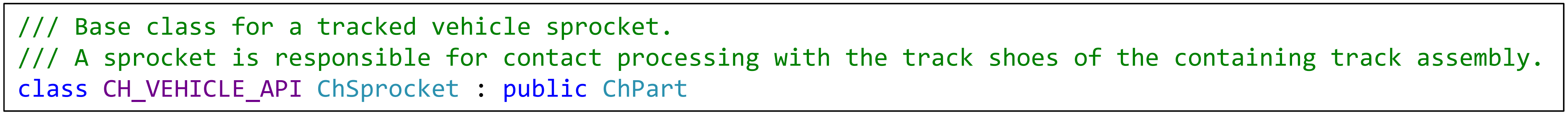
```
{
  "Name":                         "M113 Double RoadWheel Left",
  "Type":                         "RoadWheel",
  "Template":                     "DoubleRoadWheel",

  "Wheel":
  {
    "Radius":                     0.305,
    "Width":                      0.181,
    "Gap":                        0.051,
    "Mass":                       561.1,
    "Inertia":                    [19.82, 26.06, 19.82]
  },

  "Contact Material":
  {
    "Coefficient of Friction":    0.7,
    "Coefficient of Restitution": 0.1,
    "Properties": {
        "Young Modulus":          1e7,
        "Poisson Ratio":          0.3
    },
    "Coefficients": {
        "Normal Stiffness":       2e5,
        "Normal Damping":         40.0,
        "Tangential Stiffness":   2e5,
        "Tangential Damping":     20.0
    }
  },

  "Visualization":
  {
    "Mesh Filename":              "M113/Roller_L.obj",
    "Mesh Name":                  "Roller_L_POV_geom"
  }
}
```



Sprocket Subsystem

ChSprocket base class

- A sprocket is responsible for collision detection and contact processing between the sprocket and the track shoes
- A derived class which implements a particular sprocket template must specify the custom collision callback object and provide the gear profile as a 2D path.
- The gear profile, a ChLinePath geometric object, is made up of an arbitrary number of sub-paths of type ChLineArc or ChLineSegment sub-lines.
- These must be added in clockwise order, and the end of sub-path i must be coincident with beginning of sub-path i+1.



```
/// Base class for a tracked vehicle sprocket.
/// A sprocket is responsible for contact processing with the track shoes of the containing track assembly.
class CH_VEHICLE_API ChSprocket : public ChPart
```

ChSprocket class members

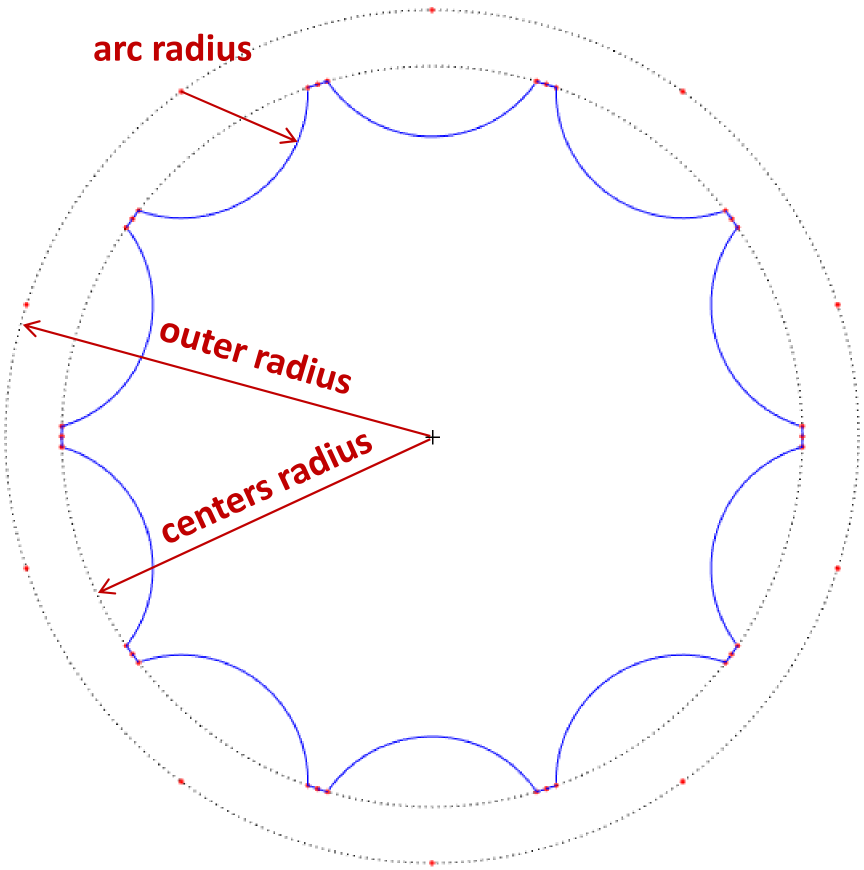
- A ChSprocket has:

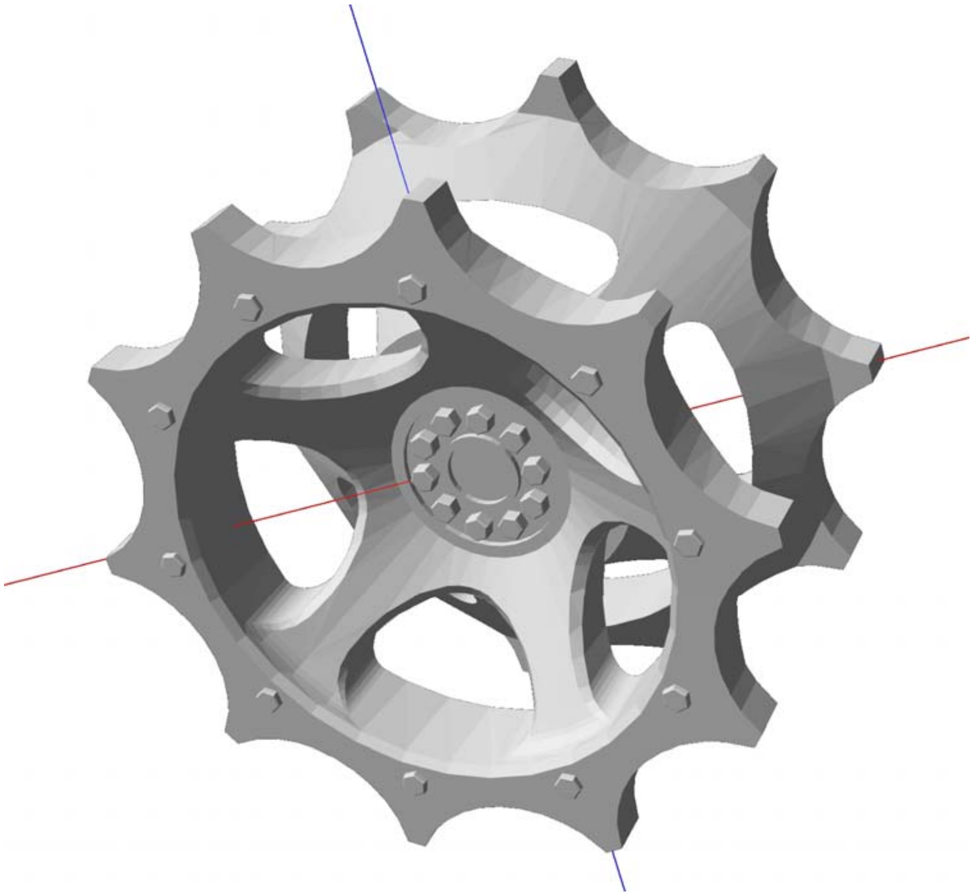
```
std::shared_ptr<ChBody> m_gear;                   ///< handle to the sprocket gear body
std::shared_ptr<ChShaft> m_axle;                  ///< handle to gear shafts
std::shared_ptr<ChShaftsBody> m_axle_to_spindle;  ///< handle to gear-shaft connector
std::shared_ptr<ChLinkLockRevolute> m_revolute;   ///< handle to sprocket revolute joint

ChSystem::ChCustomComputeCollisionCallback* m_callback;  ///< custom collision functor object

float m_friction;       ///< contact coefficient of friction
float m_restitution;    ///< contact coefficient of restitution
float m_young_modulus;  ///< contact material Young modulus
float m_poisson_ratio;  ///< contact material Poisson ratio
float m_kn;             ///< normal contact stiffness
float m_gn;             ///< normal contact damping
float m_kt;             ///< tangential contact stiffness
float m_gt;             ///< tangential contact damping
```

Sprocket Templates

Single-pin sprocket

ChSprocketSinglePin geometry





JSON specification for single-pin sprocket (1/2)

```
{
  "Name":                              "M113 SinglePin Sprocket Left",
  "Type":                              "Sprocket",
  "Template":                          "SprocketSinglePin",

  "Number Teeth":                      10,
  "Gear Mass":                         436.7,
  "Gear Inertia":                      [12.22, 13.87, 12.22],
  "Axle Inertia":                      1.0,
  "Gear Separation":                   0.225,

  "Profile":
  {
      "Addenum Radius":                0.2605,
      "Arc Radius":                    0.089,
      "Arc Centers Radius":            0.3,
      "Assembly Radius":               0.245
  },
```

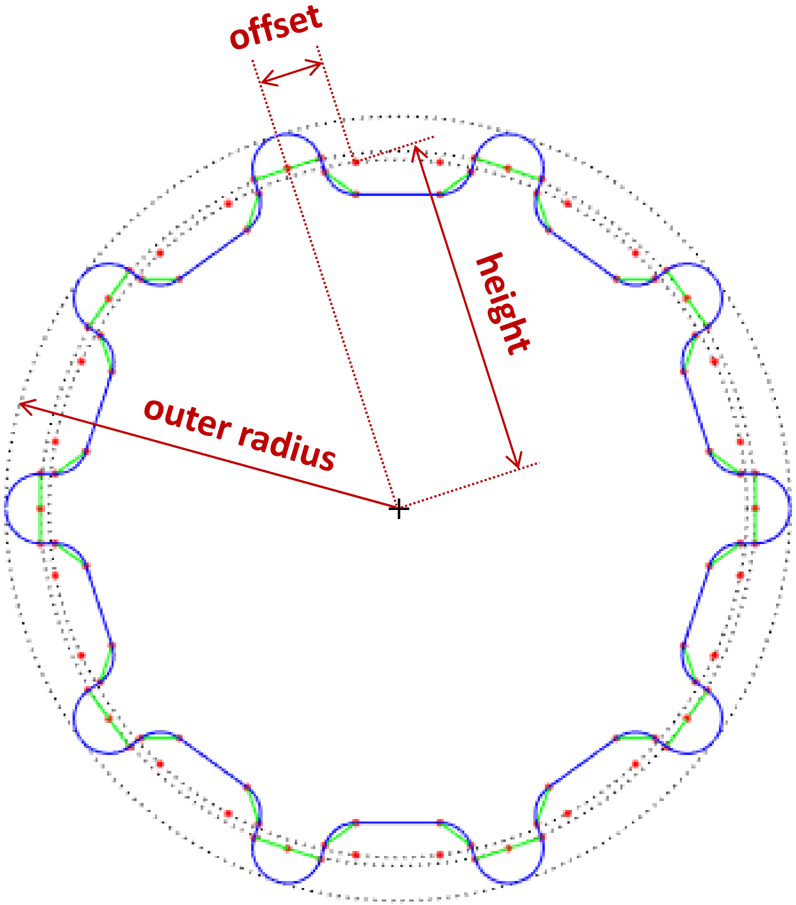
JSON specification for single-pin sprocket (2/2)

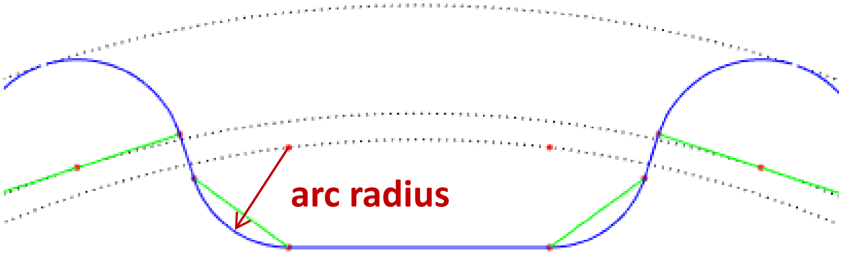
```
   "Contact Material":
   {
       "Coefficient of Friction":    0.4,
       "Coefficient of Restitution": 0.1,

       "Properties": {
           "Young Modulus":          1e7,
           "Poisson Ratio":          0.3
       },

       "Coefficients": {
           "Normal Stiffness":       2e5,
           "Normal Damping":         40.0,
           "Tangential Stiffness":   2e5,
           "Tangential Damping":     20.0
       }
   },

   "Visualization":
   {
       "Mesh Filename":              "M113/Sprocket_L.obj",
       "Mesh Name":                  "Sprocket_L_POV_geom"
   }
 }
```

Sprocket Templates

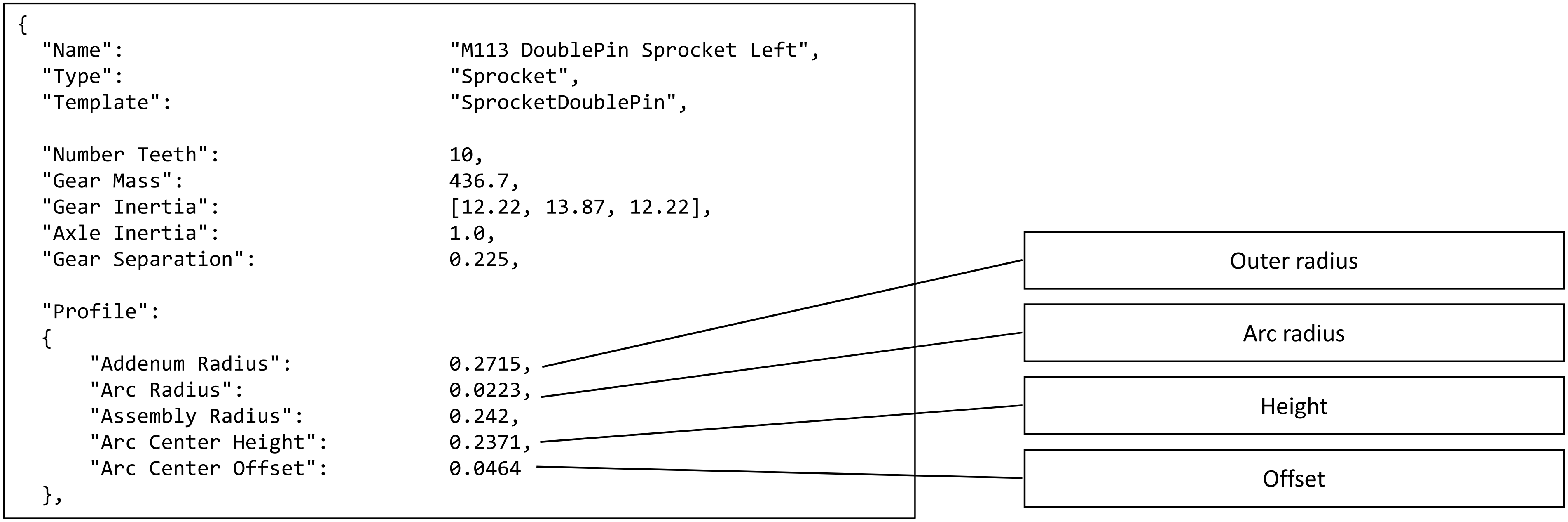
Double-pin sprocket

ChSprocketDoublePin geometry





JSON specification for double-pin sprocket (1/2)



```
{
  "Name":                              "M113 DoublePin Sprocket Left",
  "Type":                              "Sprocket",
  "Template":                          "SprocketDoublePin",

  "Number Teeth":                      10,
  "Gear Mass":                         436.7,
  "Gear Inertia":                      [12.22, 13.87, 12.22],
  "Axle Inertia":                      1.0,
  "Gear Separation":                   0.225,

  "Profile":
  {
      "Addenum Radius":                0.2715,
      "Arc Radius":                    0.0223,
      "Assembly Radius":               0.242,
      "Arc Center Height":             0.2371,
      "Arc Center Offset":             0.0464
  },
```

Outer radius

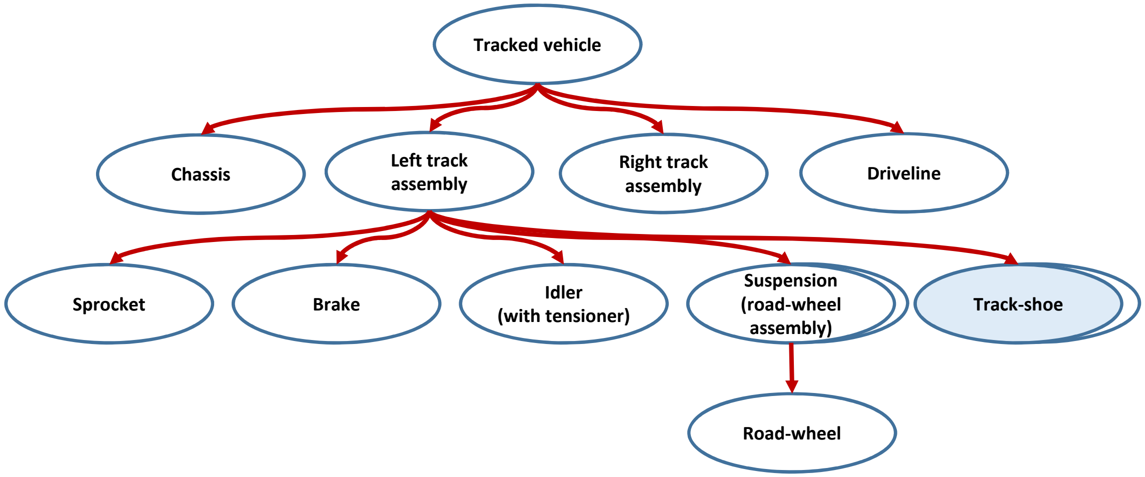
Arc radius

Height

Offset

JSON specification for double-pin sprocket (2/2)

```
  "Contact Material":
  {
      "Coefficient of Friction":    0.4,
      "Coefficient of Restitution": 0.1,

      "Properties": {
          "Young Modulus":          1e7,
          "Poisson Ratio":          0.3
      },

      "Coefficients": {
          "Normal Stiffness":       2e5,
          "Normal Damping":         40.0,
          "Tangential Stiffness":   2e5,
          "Tangential Damping":     20.0
      }
  }
}
```



Track-shoe Subsystem

ChTrackShoe base class

- Specifies the interface for the track shoe subsystem
- Provides the contact material properties
- A derived class must implement:
    - a method to initialize the track shoe subsystem at a given location and with a given orientation
    - a method to connect two adjacent track shoes (always assumed to have proper relative positions)

```
/// Base class for a track shoe.
class CH_VEHICLE_API ChTrackShoe : public ChPart
```

ChTrackShoe class members

- A ChTrackShoe has:

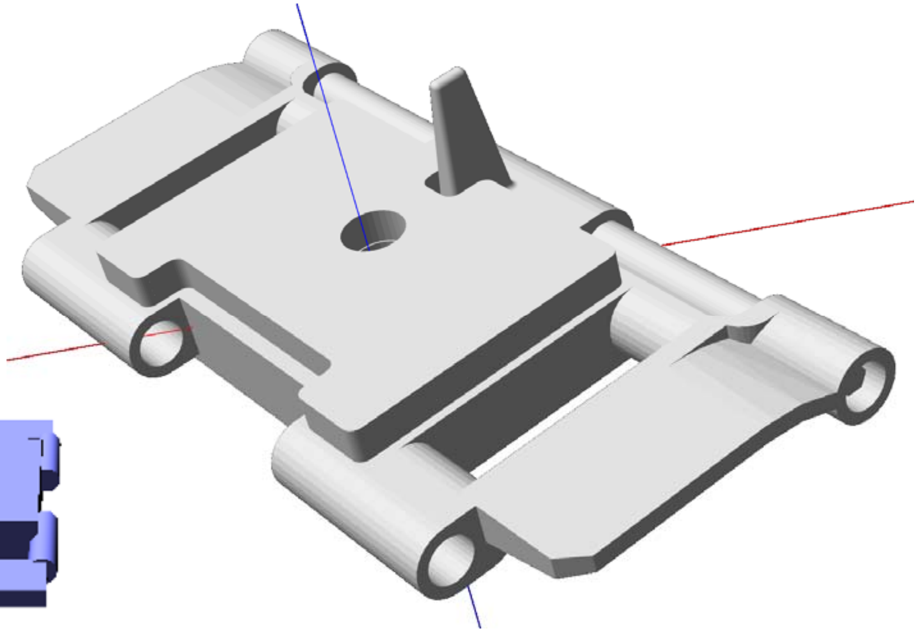
```
size_t m_index;              ///< index of this track shoe within its containing track assembly
std::shared_ptr<ChBody> m_shoe;  ///< handle to the shoe body

float m_friction;       ///< contact coefficient of friction
float m_restitution;    ///< contact coefficient of restitution
float m_young_modulus;  ///< contact material Young modulus
float m_poisson_ratio;  ///< contact material Poisson ratio
float m_kn;             ///< normal contact stiffness
float m_gn;             ///< normal contact damping
float m_kt;             ///< tangential contact stiffness
float m_gt;             ///< tangential contact damping
```

Track-shoe Templates

Single-pin track-shoe

ChTrackShoeSinglePin geometry

- Single-pin, single-body track shoe
- Central guiding pin (i.e. consistent with ChDoubleIdler, ChDoubleRoadWheel)
- Connection to adjacent track shoe is through revolute joints (except the track loop closure)



JSON specification for single-pin track-shoe (1/2)

```
{
  "Name":                           "M113 SinglePin TrackShoe Left",
  "Type":                           "TrackShoe",
  "Template":                       "TrackShoeSinglePin",

  "Shoe":
  {
      "Height":                     0.06,
      "Pitch":                      0.154,
      "Mass":                       18.02,
      "Inertia":                    [0.22, 0.04, 0.25]
  },

  "Contact Geometry":
  {
      "Shoe":
      {
          "Pad Dimensions":         [0.11, 0.19, 0.06],
          "Pad Location":           [0, 0, 0],
          "Guide Dimensions":       [0.0284, 0.0114, 0.075],
          "Guide Location":         [0.045, 0, 0.0375]
      },

      "Cylinder":
      {
          "Radius":                 0.015,
          "Front Offset":           0.0535,
          "Rear Offset":            -0.061
      }
  },
```

JSON specification for single-pin track-shoe (2/2)

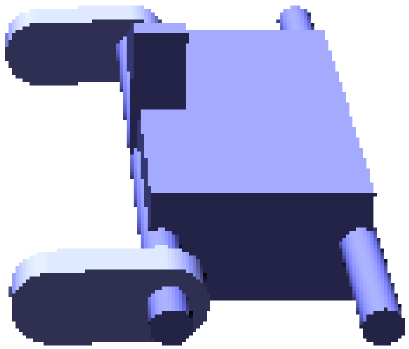
```
  "Contact Material":
  {
      "Coefficient of Friction":    0.8,
      "Coefficient of Restitution": 0.1,

      "Properties": {
          "Young Modulus":          1e7,
          "Poisson Ratio":          0.3
      },

      "Coefficients": {
          "Normal Stiffness":       2e5,
          "Normal Damping":         40.0,
          "Tangential Stiffness":   2e5,
          "Tangential Damping":     20.0
      }
  },

  "Visualization":
  {
      "Mesh Filename":              "M113/TrackShoe.obj",
      "Mesh Name":                  "TrackShoe_POV_geom"
  }
}
```

Track-shoe Templates

Double-pin track-shoe

ChTrackShoeDoublePin geometry

- Double-pin, single-body track shoe
- Central guiding pin (i.e. consistent with ChDoubleIdler, ChDoubleRoadWheel)
- Connection to adjacent track shoe is through spherical joints (except the track loop closure)
- Revolute joints between shoe body and connector bodies

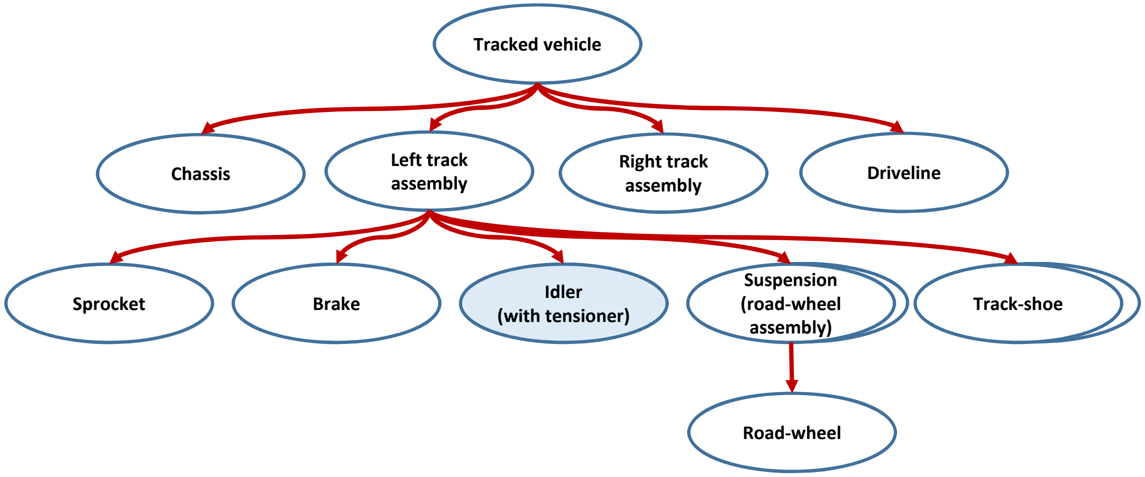


JSON specification for double-pin track-shoe (1/2)

```
{
  "Name":                           "M113 DoublePin TrackShoe Left",
  "Type":                           "TrackShoe",
  "Template":                       "TrackShoeDoublePin",

  "Shoe":
  {
      "Length":                     0.0984,
      "Width":                      0.2781,
      "Height":                     0.06,
      "Mass":                       18.02,
      "Inertia":                    [0.22, 0.04, 0.25]
  },

  "Connector":
  {
      "Radius":                     0.02,
      "Length":                     0.054,
      "Width":                      0.02,
      "Mass":                       2.0,
      "Inertia":                    [0.1, 0.1, 0.1]
  },

  "Contact Geometry":
  {
      "Shoe":
      {
          "Pad Dimensions":         [0.11, 0.19, 0.06],
          "Pad Location":           [0, 0, 0],
          "Guide Dimensions":       [0.0284, 0.0114, 0.075],
          "Guide Location":         [0.045, 0, 0.0375]
      }
 },
```

JSON specification for double-pin track-shoe (2/2)

```
  "Contact Material":
  {
      "Coefficient of Friction":    0.8,
      "Coefficient of Restitution": 0.1,

      "Properties": {
          "Young Modulus":          1e7,
          "Poisson Ratio":          0.3
      },

      "Coefficients": {
          "Normal Stiffness":       2e5,
          "Normal Damping":         40.0,
          "Tangential Stiffness":   2e5,
          "Tangential Damping":     20.0
      }
  }
}
```



Idler Subsystem

ChIdler base class

- An idler subsystem consists of the idler wheel and a connecting body.
- The idler wheel is connected through a revolute joint to the connecting body which in turn is connected to the chassis through a translational joint.
- A linear actuator acts as a tensioner.

- An idler subsystem is defined with respect to a frame centered at the origin of the idler wheel, possibly pitched relative to the chassis reference frame.
- The translational joint is aligned with the x axis of this reference frame, while the axis of rotation of the revolute joint is aligned with its y axis.

```
/// Base class for an idler subsystem.
/// An idler consists of the idler wheel and a connecting body.  The idler wheel is connected
/// through a revolute joint to the connecting body which in turn is connected to the chassis
/// through a translational joint. A linear actuator acts as a tensioner.
class CH_VEHICLE_API ChIdler : public ChPart
```

ChIdler class members

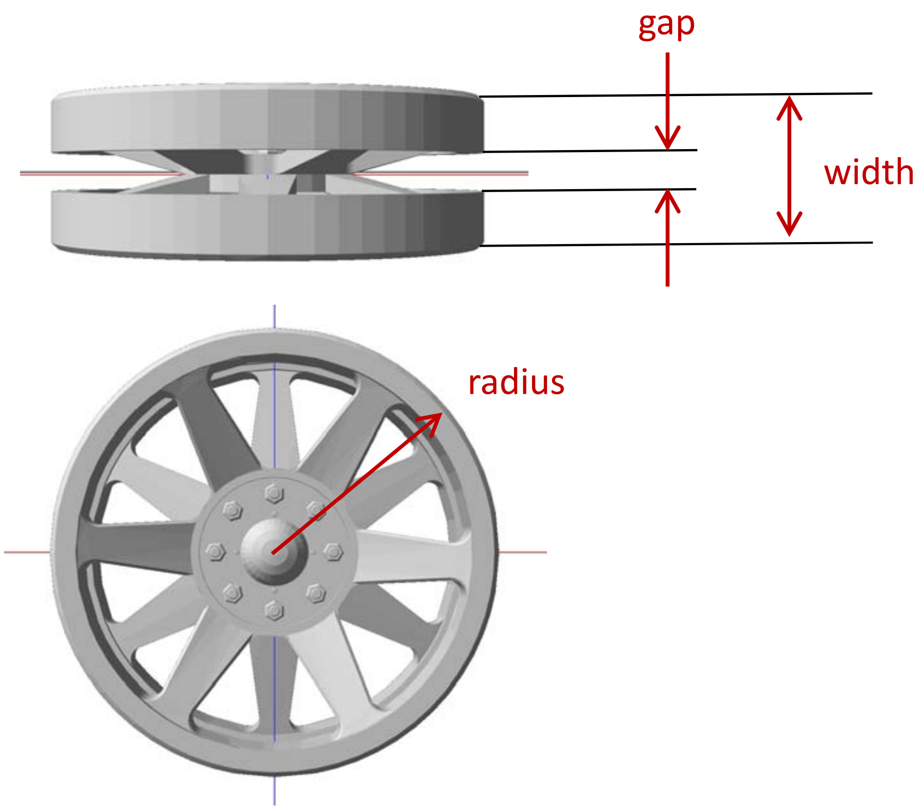
- A ChIdler has:

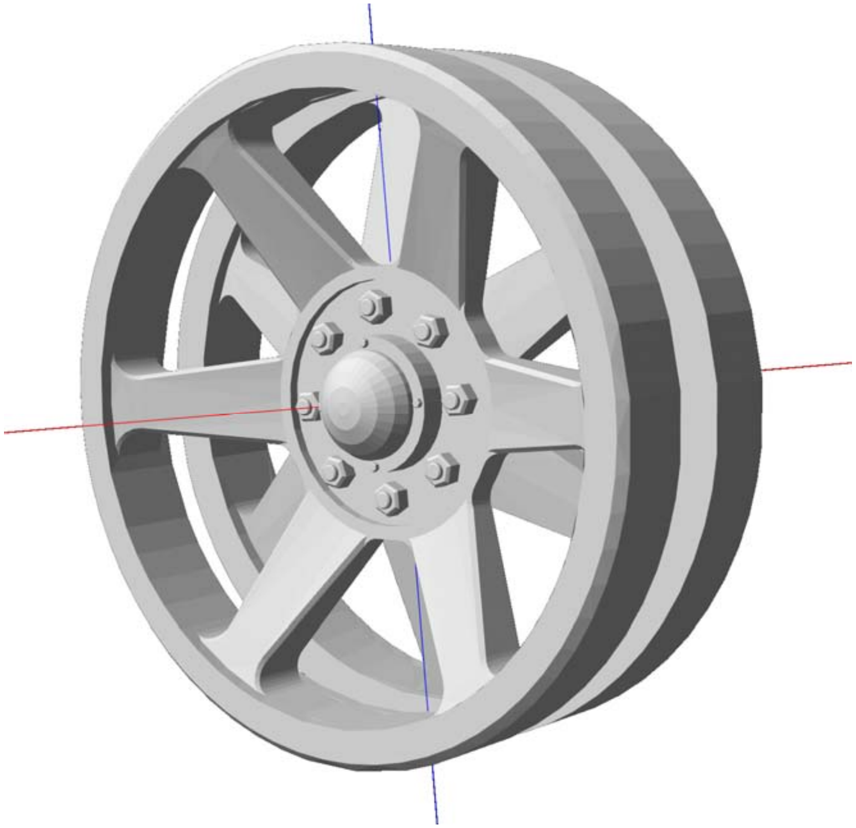
```
std::shared_ptr<ChBody> m_wheel;                    ///< handle to the idler wheel body
std::shared_ptr<ChBody> m_carrier;                  ///< handle to the carrier body
std::shared_ptr<ChLinkLockRevolute> m_revolute;     ///< handle to wheel-carrier revolute joint
std::shared_ptr<ChLinkLockPrismatic> m_prismatic;   ///< handle to carrier-chassis translational joint
std::shared_ptr<ChLinkSpringCB> m_tensioner;        ///< handle to the TSDA tensioner element

float m_friction;       ///< contact coefficient of friction
float m_restitution;    ///< contact coefficient of restitution
float m_young_modulus;  ///< contact material Young modulus
float m_poisson_ratio;  ///< contact material Poisson ratio
float m_kn;             ///< normal contact stiffness
float m_gn;             ///< normal contact damping
float m_kt;             ///< tangential contact stiffness
float m_gt;             ///< tangential contact damping
```

Idler Templates

Double idler

ChDoubleIdler geometry



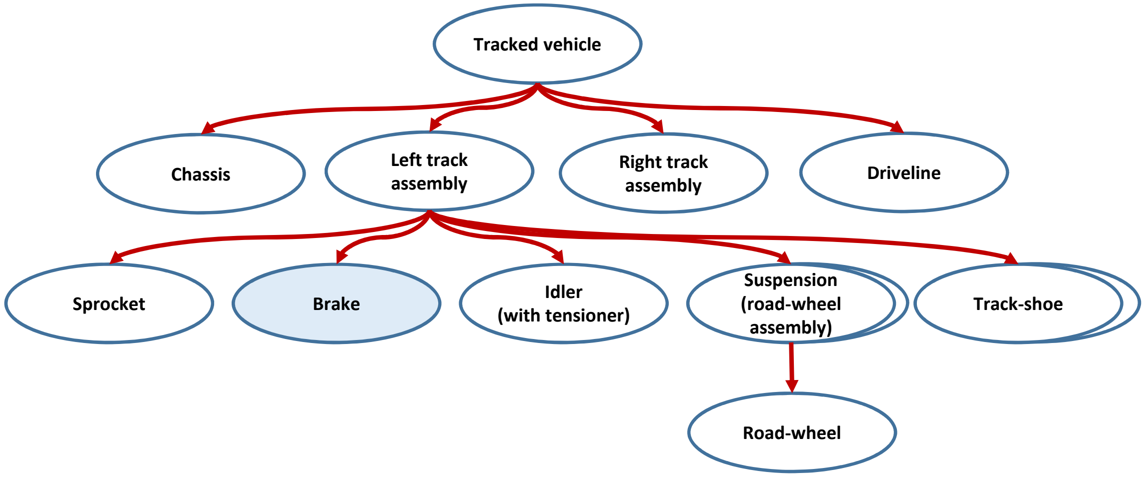


JSON specification for double idler (1/2)

```
{
  "Name":                             "M113 Double Idler Left",
  "Type":                             "Idler",
  "Template":                         "DoubleIdler",

  "Wheel":
  {
      "Radius":                       0.255,
      "Width":                        0.181,
      "Gap":                          0.051,
      "Mass":                         429.5,
      "COM":                          [0, 0, 0],
      "Inertia":                      [12.55, 14.70, 12.55]
  },

  "Carrier":
  {
      "Mass":                         50.0,
      "COM":                          [0, -0.1, 0],
      "Inertia":                      [2, 2, 2],
      "Location Chassis":             [0, -0.2, 0],
      "Visualization Radius":         0.02,
      "Pitch Angle":                  0
  },

  "Tensioner":
  {
    "Location Carrier":               [0, -0.2, 0],
    "Location Chassis":               [0.5, -0.2, 0],
    "Preload":                        2e4,
    "Free Length":                    0.75,
    "Spring Coefficient":             1e6,
    "Damping Coefficient":            1.4e4
  },
```

JSON specification for double idler (2/2)

```
  "Contact Material":
  {
      "Coefficient of Friction":    0.7,
      "Coefficient of Restitution": 0.1,

      "Properties": {
          "Young Modulus":          1e8,
          "Poisson Ratio":          0.3
      },

      "Coefficients": {
          "Normal Stiffness":       2e5,
          "Normal Damping":         40.0,
          "Tangential Stiffness":   2e5,
          "Tangential Damping":     20.0
      }
  },

  "Visualization":
  {
      "Mesh Filename":              "M113/Idler_L.obj",
      "Mesh Name":                  "Idler_L_POV_geom"
  }
}
```



Brake Subsystem

ChTrackBrake base class

- Defines the common interface for any brake subsystem
- All classes defining particular brake templates inherit from ChTrackBrake

```
///
/// Base class for a track brake subsystem
///
class CH_VEHICLE_API ChTrackBrake : public ChPart
```

Brake Templates

Simple track brake

ChTrackBrakeSimple

- Simple brake model using a constant torque opposing sprocket rotation.
- Uses a speed-dependent torque
- It cannot simulate sticking
- On initialization, it is associated with a revolute joint connecting the sprocket gear body
- Has a single parameter, the maximum braking torque

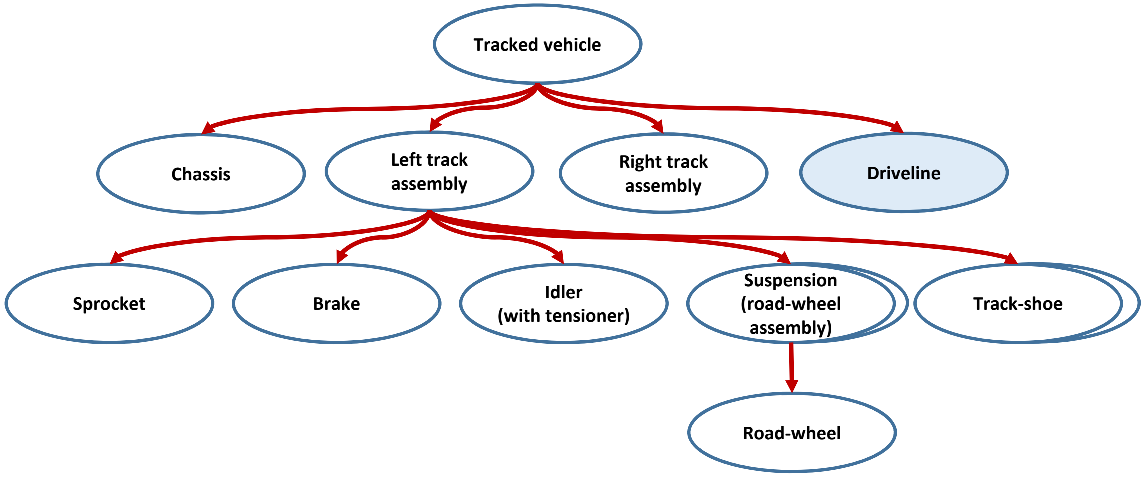
JSON specification file for TrackBrakeSimple

```
{
  "Name":                          "M113 Siple Brake",
  "Type":                          "TrackBrake",
  "Template":                      "TrackBrakeSimple",

  "Maximum Torque":                10000
}
```

Subsystem type (string)

Template type (string)

Maximum braking torque in Nm (double)



Driveline Subsystem

ChTrackDriveline base class

```
/// Base class for a tracked vehicle driveline.
class CH_VEHICLE_API ChTrackDriveline : public ChPart
```

Driveline Templates

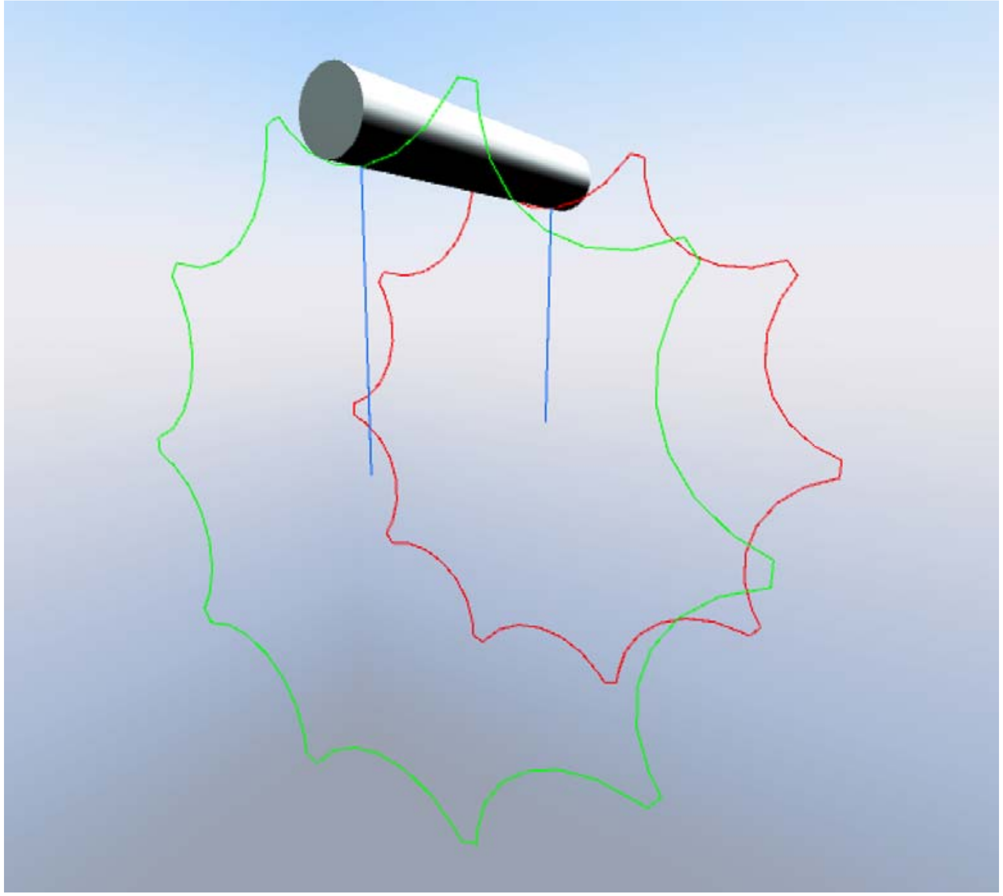
Simple driveline

JSON specification for simple track driveline

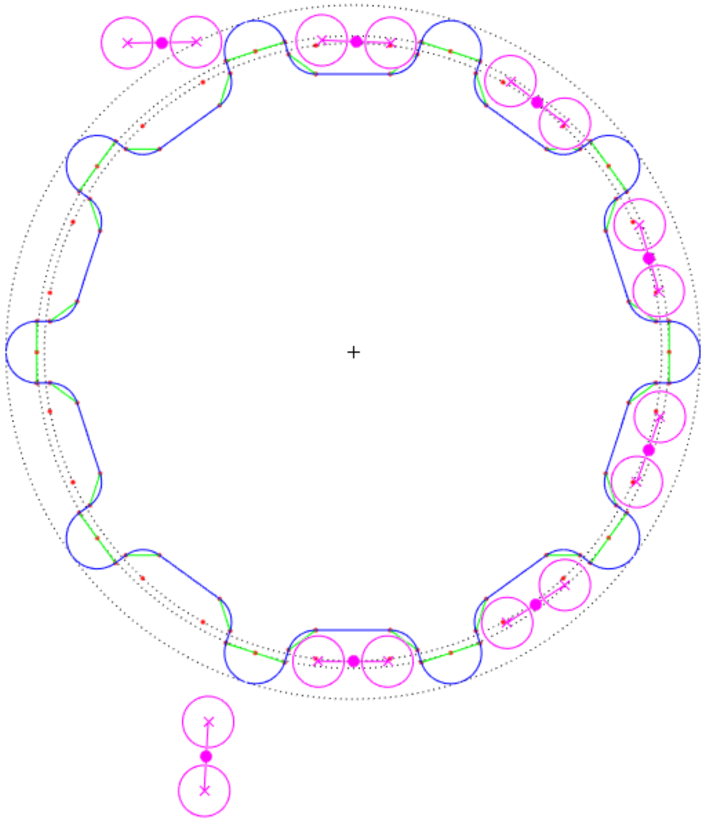
```
{
  "Name":                         "M113 Simple Driveline",
  "Type":                         "TrackDriveline",
  "Template":                     "SimpleTrackDriveline",

  "Differential Max Bias":      1.0
}
```

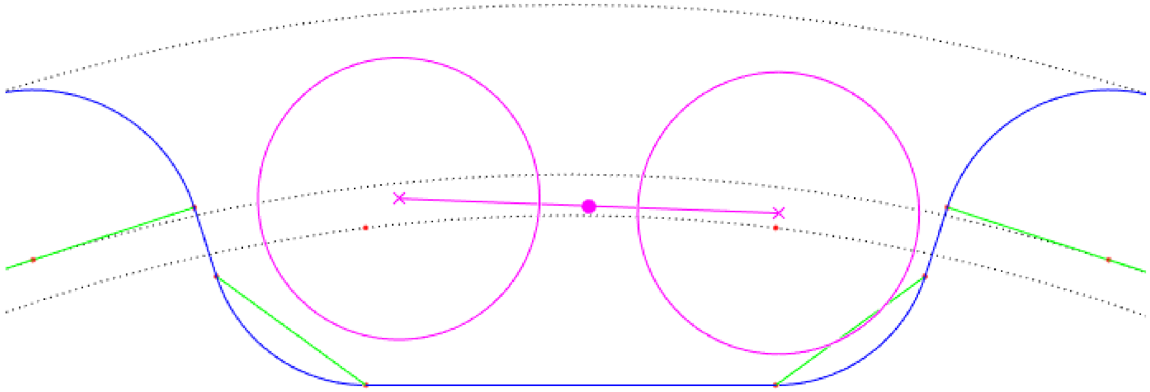
Contact processing and monitoring

Sprocket – track shoe (single-pin)



Sprocket – track shoe (double-pin)





Contact monitoring

- A ChTrackVehicle embeds a contact monitoring object of type ChTrackContactManager
- Maintains lists of contacts on the two sprockets, two idler wheels, and one track shoe from each track assembly

```
/// Class for monitoring contacts of tracked vehicle subsystems.
class ChTrackContactManager : public chrono::ChReportContactCallback {
  public:
    ChTrackContactManager();

    void MonitorContacts(int flags) { m_flags |= flags; }
    void SetContactCollection(bool val) { m_collect = val; }
    void WriteContacts(const std::string& filename);

    void SetTrackShoeIndexLeft(size_t idx) { m_shoe_index_L = idx; }
    void SetTrackShoeIndexRight(size_t idx) { m_shoe_index_R = idx; }

    void Process(ChTrackedVehicle* vehicle);
```

Enabling contact monitoring

- ChTrackedVehicle methods:

```
/// Set contacts to be monitored.
/// Contact information will be tracked for the specified subsystems.
void MonitorContacts(int flags) { m_contacts->MonitorContacts(flags); }

/// Turn on/off contact data collection.
/// If enabled, contact information will be collected for all monitored subsystems.
void SetContactCollection(bool val) { m_contacts->SetContactCollection(val); }
```

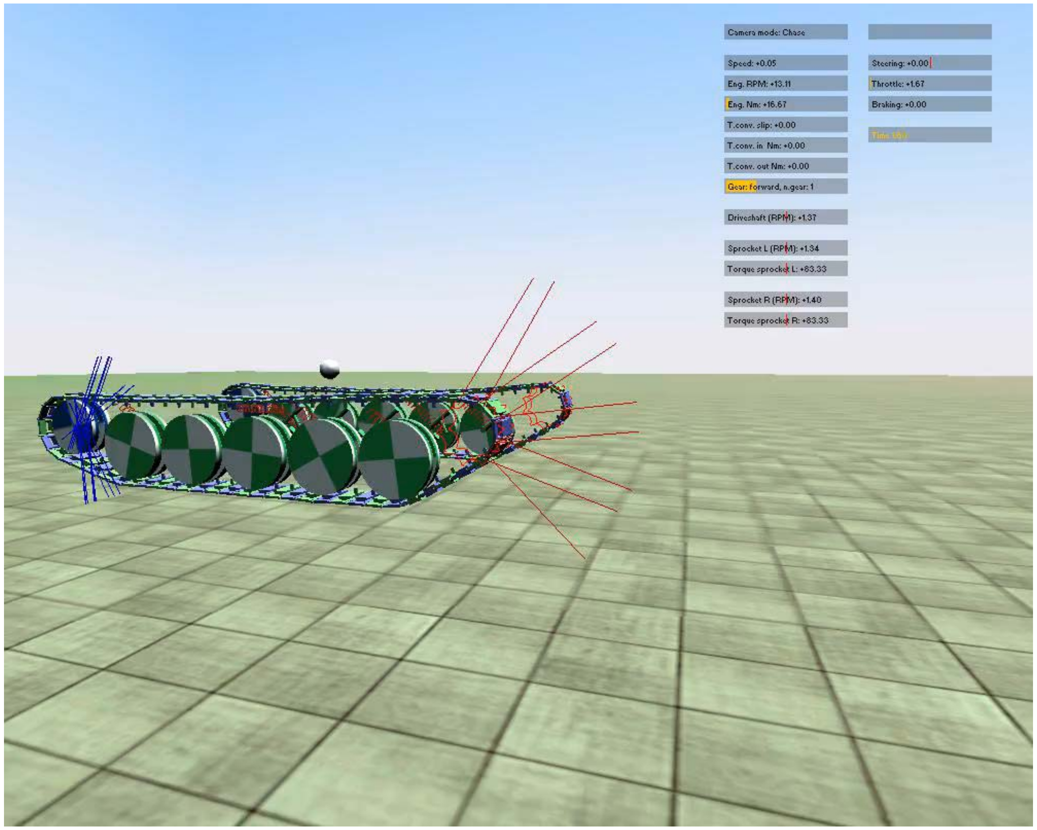
- Example (flags can be OR-ed):

```
vehicle.MonitorContacts(TrackCollide::SPROCKET_LEFT | TrackCollide::SHOES_LEFT | TrackCollide::IDLER_LEFT);
vehicle.SetContactCollection(true);
```

- Available flags:
  `SPROCKET_LEFT, SPROCKET_RIGHT, IDLER_LEFT, IDLER_RIGHT, SHOES_LEFT, SHOES_RIGHT`

Monitoring contacts

- If enabled, contacts for the specified subsystems are rendered at run-time (Irrlicht):



```
Camera mode: Chase

Speed: +0.05                  Steering: +0.00
Eng. RPM: +13.11              Throttle: +1.67
Eng. Nm: +16.67               Braking: +0.00
T.conv. slip: +0.00
T.conv. in  Nm: +0.00         Time: [illegible]
T.conv. out Nm: +0.00
Gear: forward, n.gear: 1

Driveshaft (RPM): +1.37

Sprocket L (RPM): +1.34
Torque sprocket L: +83.33

Sprocket R (RPM): +1.40
Torque sprocket R: +83.33
```

(slowed down 3x)

Monitoring contacts

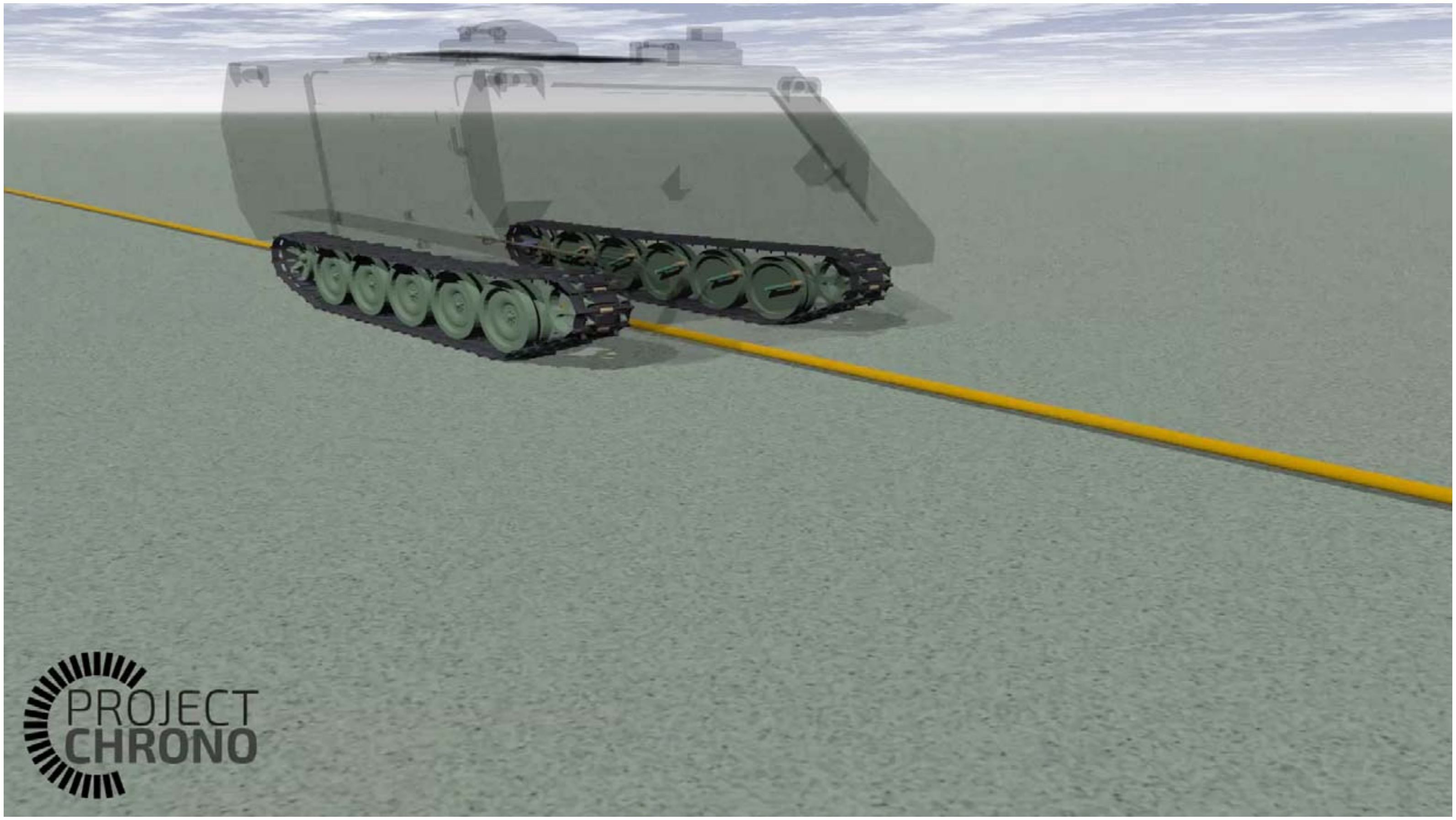
- If data collection was enabled, contact information can be written to an output file

```
/// Write contact information to file.
/// If data collection was enabled and at least one subsystem is monitored,
/// contact information is written (in CSV format) to the specified file.
void WriteContacts(const std::string& filename) { m_contacts->WriteContacts(filename); }
```

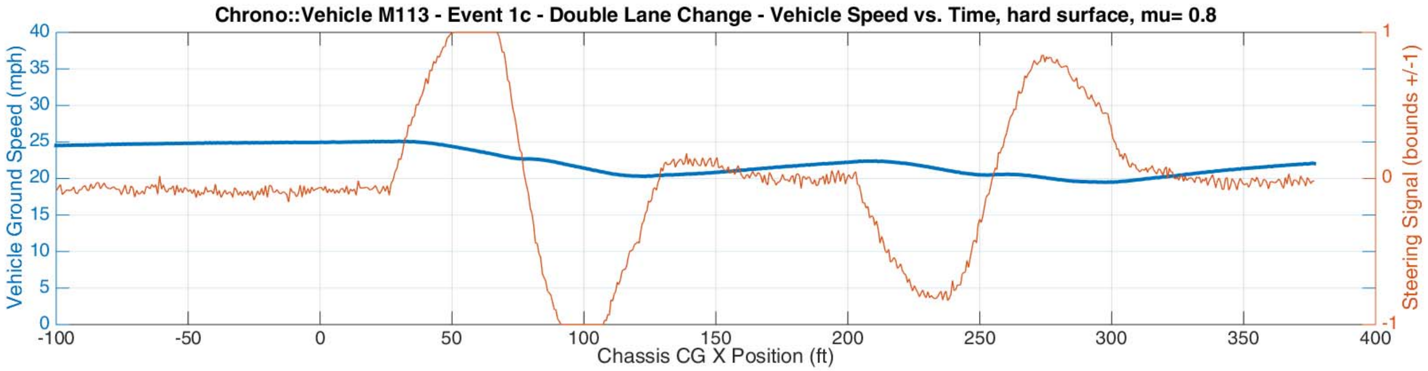
- Note: output not complete right now (WIP)

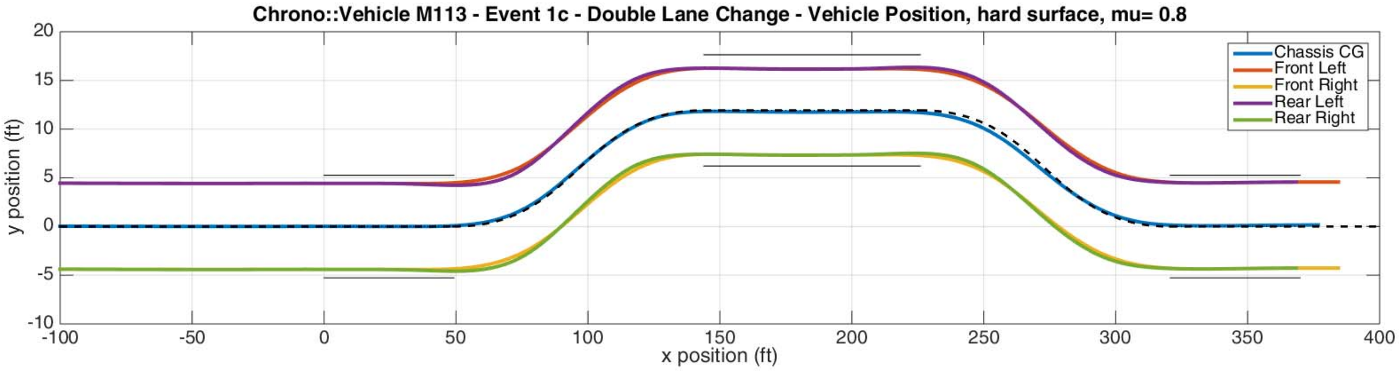
Sample simulations

M113 double-lane change (rigid terrain)

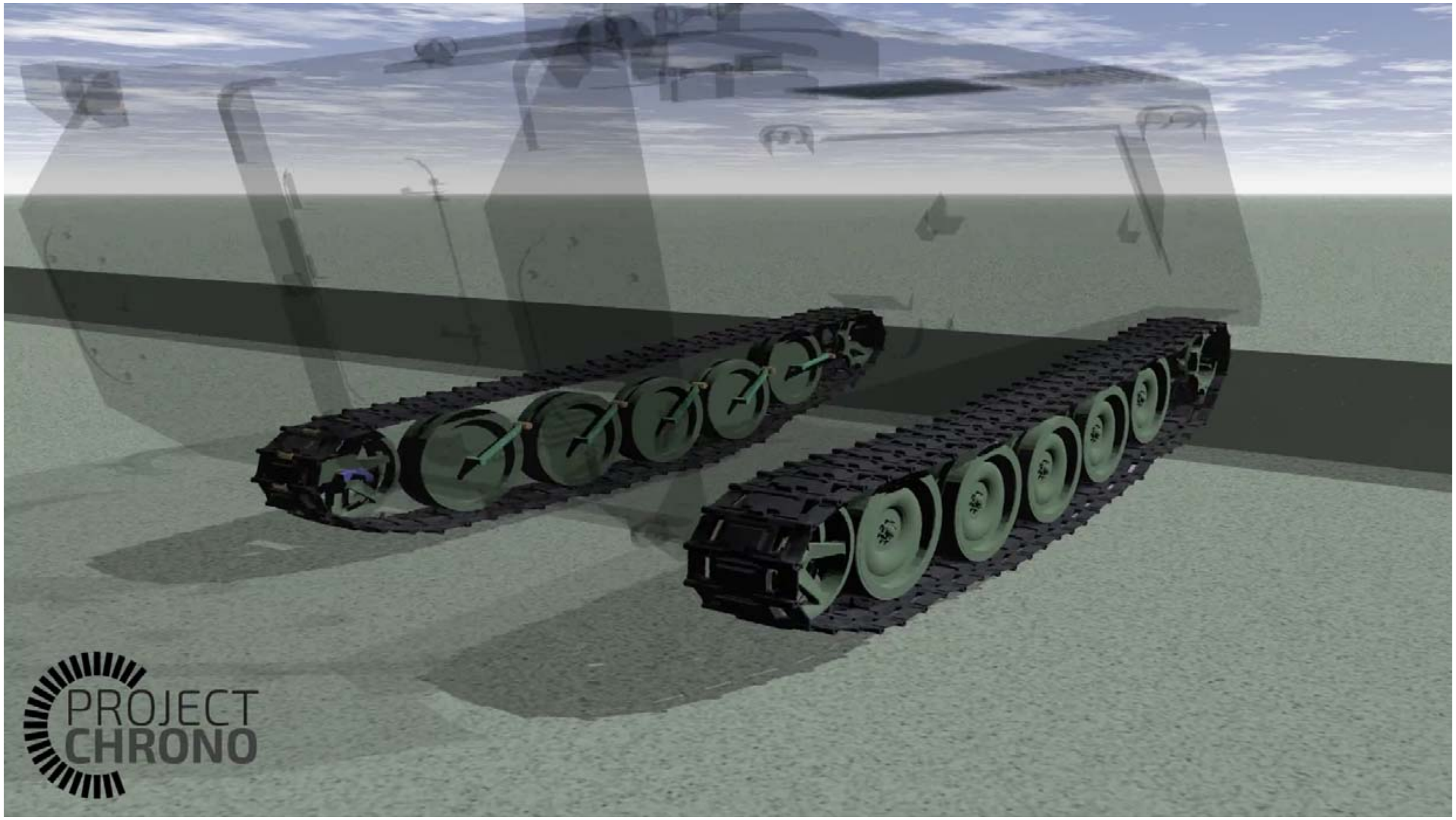


M113 double-lane change (rigid terrain)

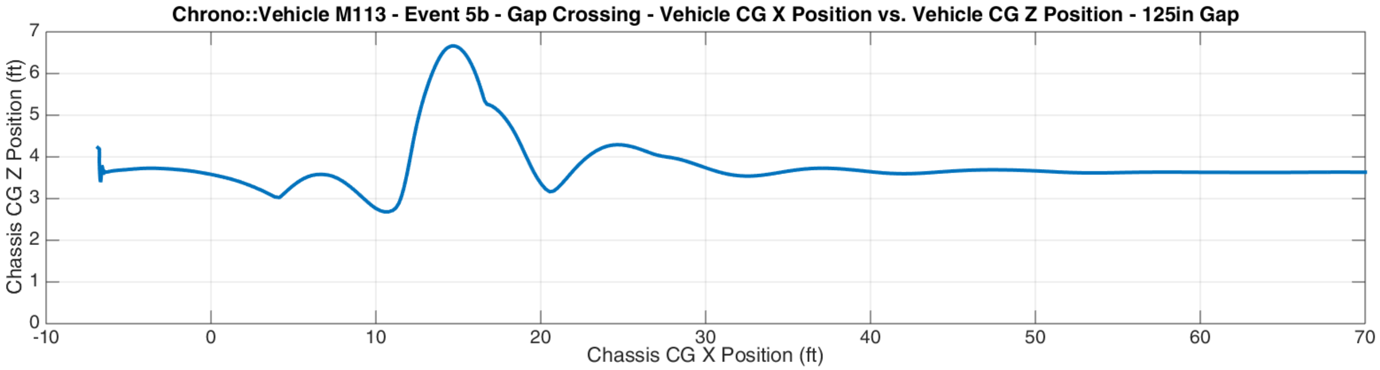

Chrono::Vehicle M113 - Event 1c - Double Lane Change - Vehicle Speed vs. Time, hard surface, mu= 0.8

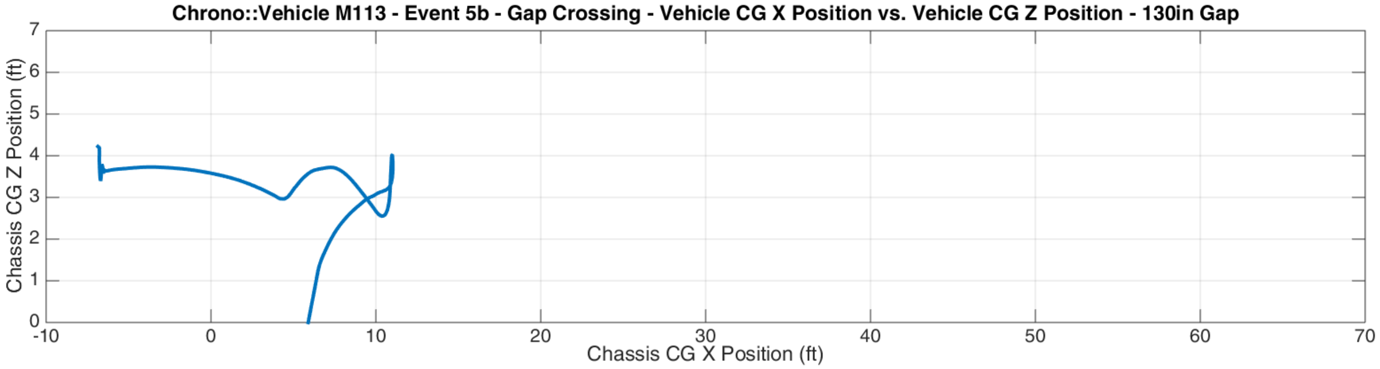

Chrono::Vehicle M113 - Event 1c - Double Lane Change - Vehicle Position, hard surface, mu= 0.8

M113 step climbing

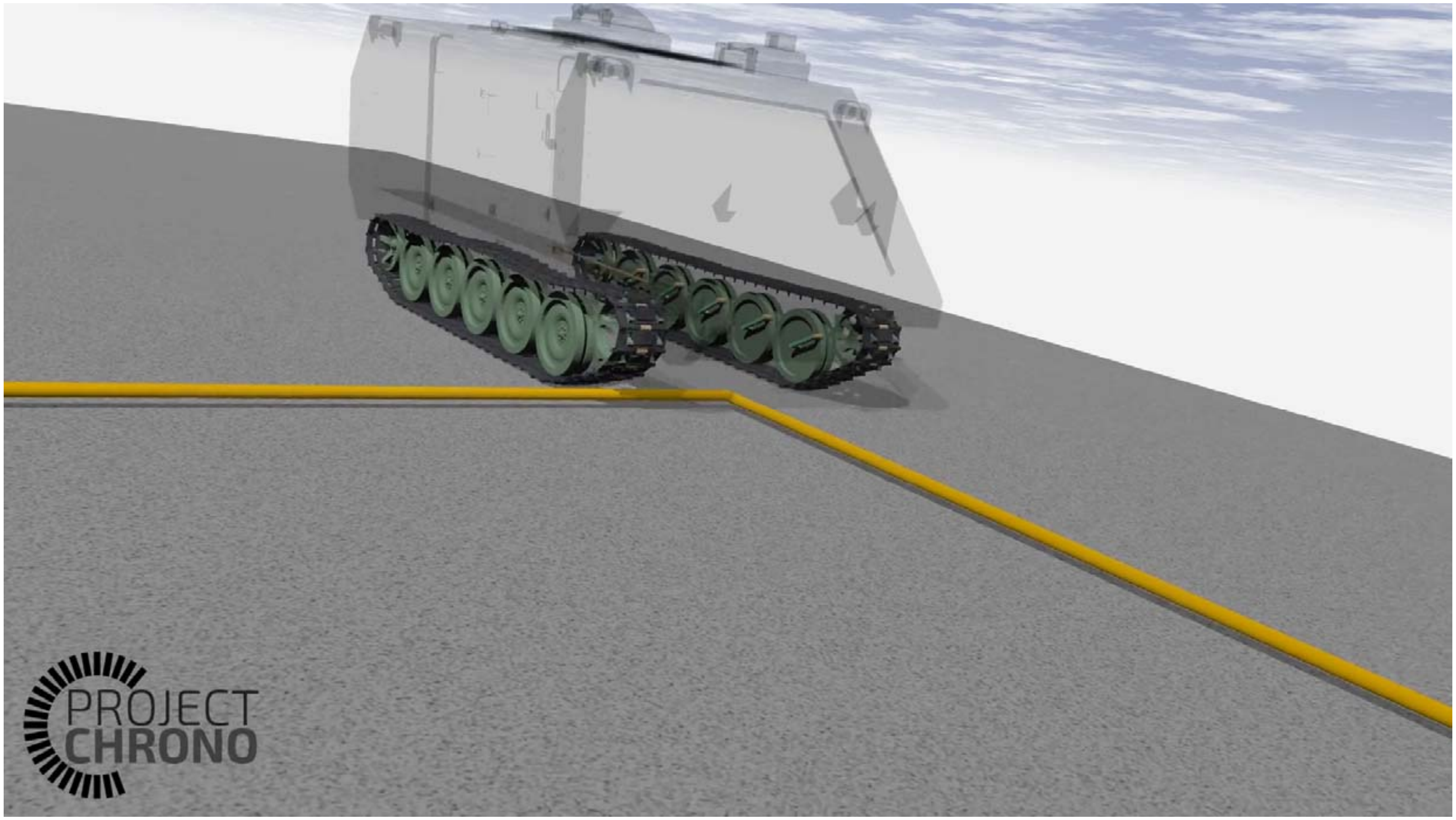


M113 step climbing

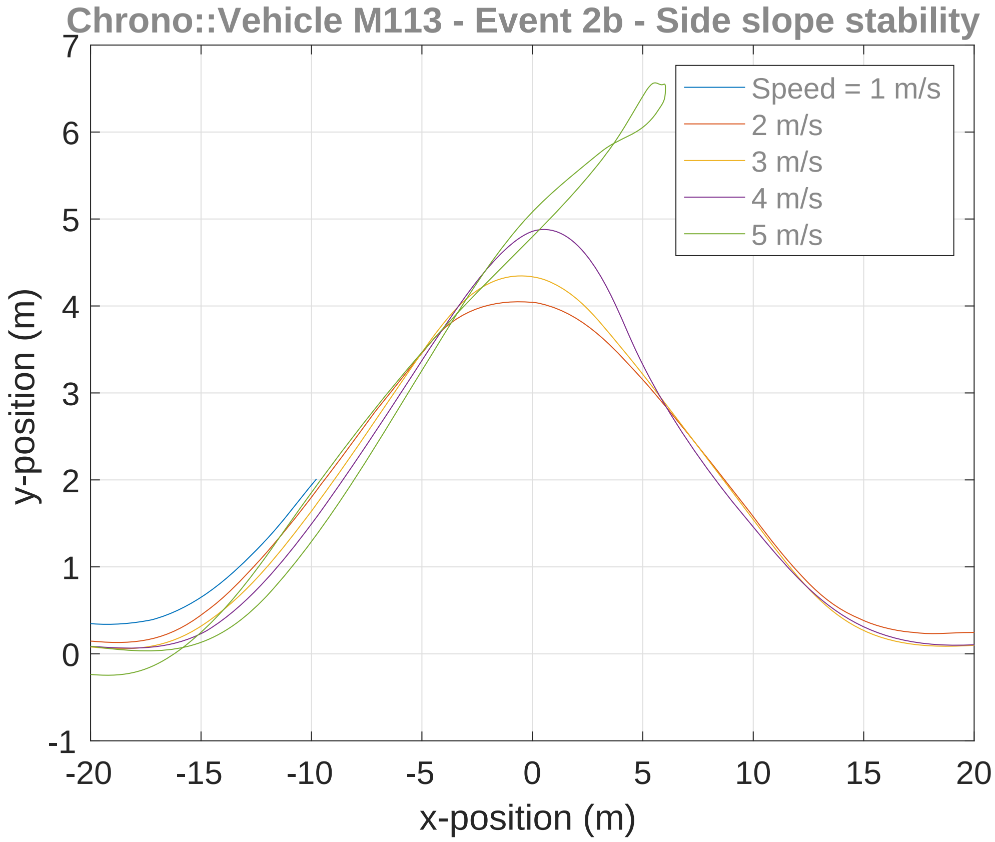




M113 slide slope object avoidance (SCM terrain)



M113 slide slope object avoidance (SCM terrain)


Chrono::Vehicle M113 - Event 2b - Side slope stability
Speed = 1 m/s
2 m/s
3 m/s
4 m/s
5 m/s
y-position (m)
x-position (m)