Advanced Rigging on Maya

Presenter - Hyun Seung Kim

Who Am I ?

- Creature TD
  - Creatures and Digital Double Dev for Visual Effect Films
  - Variety of Simulation and Rigging
- Character TD
  - Rigging Tool and Functionality Design Dev For Feature Animations
  - Character System Design and Animation System Dev
- Technical Artist
  - Design and Dev on Character Pipeline and Effects for Game
  - Dev on Environment Asset Pipeline
- Technical Art Director
  - Design workflow and solution on asset creation process on both character and environments
  - Lighting and Rendering Tech

About Advanced Rigging

- What is?
  - Two Big Categories
    i. *Character Asset Creation Process and Automations - Tool / Pipeline Building*
    ii. Functionality and Complexity for character system - Rigging

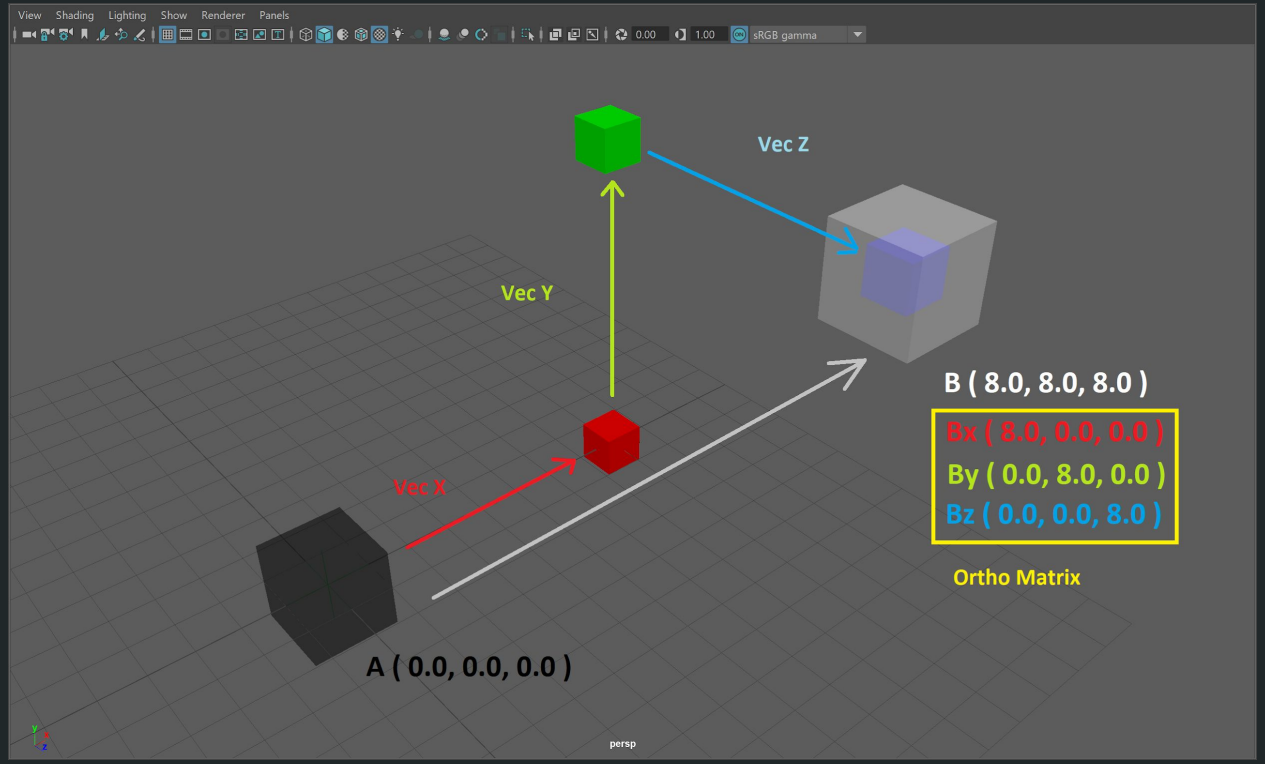
- Focus on (with Limited Time)
  - Functionality and Complexity for character system - Rigging
    i. Common Knowledge on Math
    ii. Using Node Networks to build

Rigging

- Not Crazy Math (we learned from Middle school)
  - Vector
    i. Point vs Vector
    ii. Has 2 types of Data
      - Direction (called Unit Vector)
      - Length (called Magnitude)
      - Vector = Direction * Length ➜ Unit Vector (Direction) = Vector/Length
  - Matrix
    i. Not about Movie, “red pill or blue pill”
    ii. It is simple collection of data with some rules
      - Stacked data with 4 vectors (3X3 and 4X4 Matrix)
    iii. “Matrix Transformation is Multiplication” - “*MultMatrix*” not “*AddMatrix*”

Rigging

- Visualization
- Vector
- Matrix



Rigging

- Not Crazy Math (we learned from Middle school)
  - Vector Math
    i. “VectorProduct”
      - Dot (float)
        - **Using for Checking on Direction/Amounts**
        - If Dot > 0 means input 2 Vectors are the same direction
        - If Dot == 0 means input 2 Vectors are 90 degree direction
        - If Dot < 0 means input 2 Vectors are the opposite direction
      - Cross (vector)
        - **Using for Building Matrix often**
        - Maya use LeftHand Order
        - Given Input 2 unit vector, get 3rd vector are 90 degree to inputs
        - Output Vector might **Not** be unit vector

Rigging

- Nodes?
  - Advantage
    i. Node is Small Function
    ii. Already Lots of Them on Maya
    iii. Good User Interface
    iv. Quick Prototypes of Complexity on Rigging Behaviors
    v. Custom design and Scalability
    vi. Good Documentation
  - Disadvantage
    i. Steep Running Curve
    ii. Depending on how to use, Difficult or Easier to debug
    iii. Require Math knowledge and deeper understanding on maya

Rigging

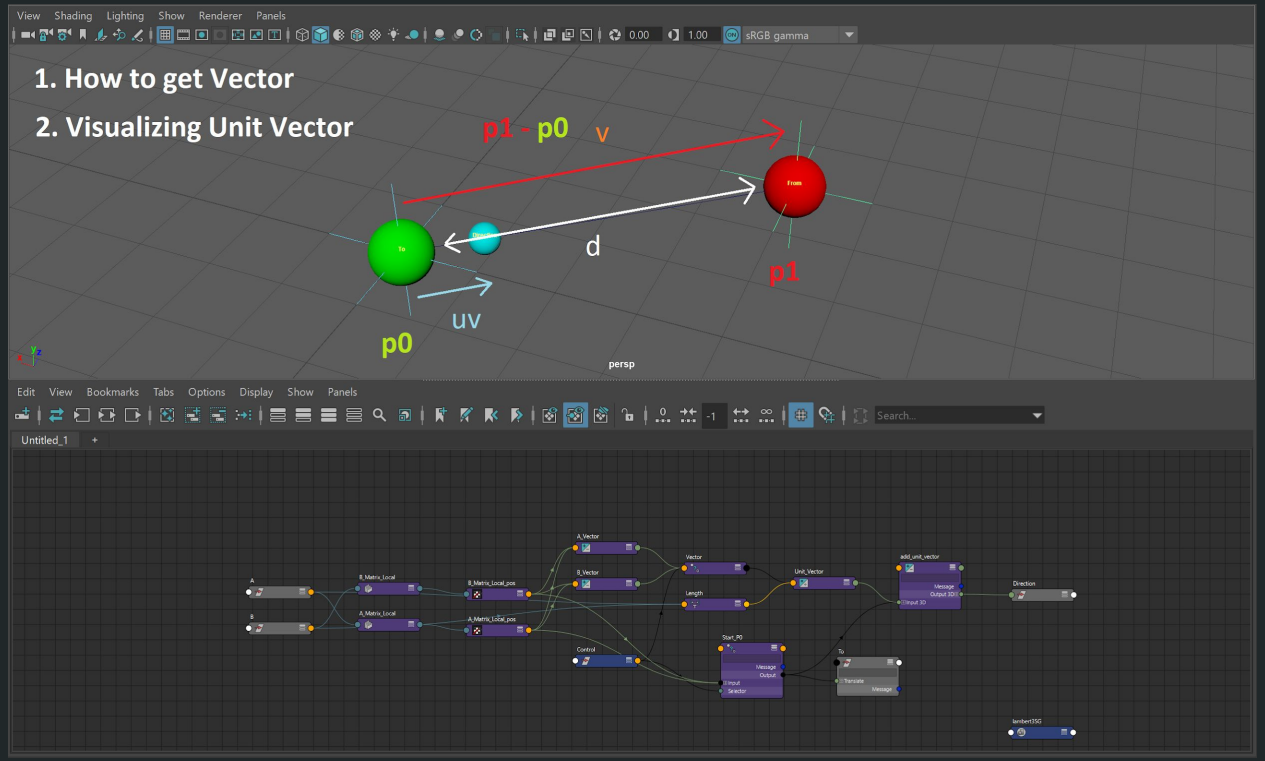
- How to learn
  - **Learn from Examples**
  - **Focus on “Understanding” and Not on “Memorizing”**
  - **Hands on Examples**
  - **Building your own samples**
  - Familiarity of Documentation

Rigging

- Examples (Scenes)
  - vector.ma
  - matrix.ma
  - interpolation.ma
  - space_conversion.ma
  - projectionVector_customConstraint.ma
  - recalculate_rotation_by_JointAxisOrder.ma
  - groundDection.ma
  - More complexed ones for studying
    i. foot_rocker.ma
    ii. matrixSumByWeight.ma
    iii. fnt_quadleg.ma
    iv. tailsystem_demo.ma

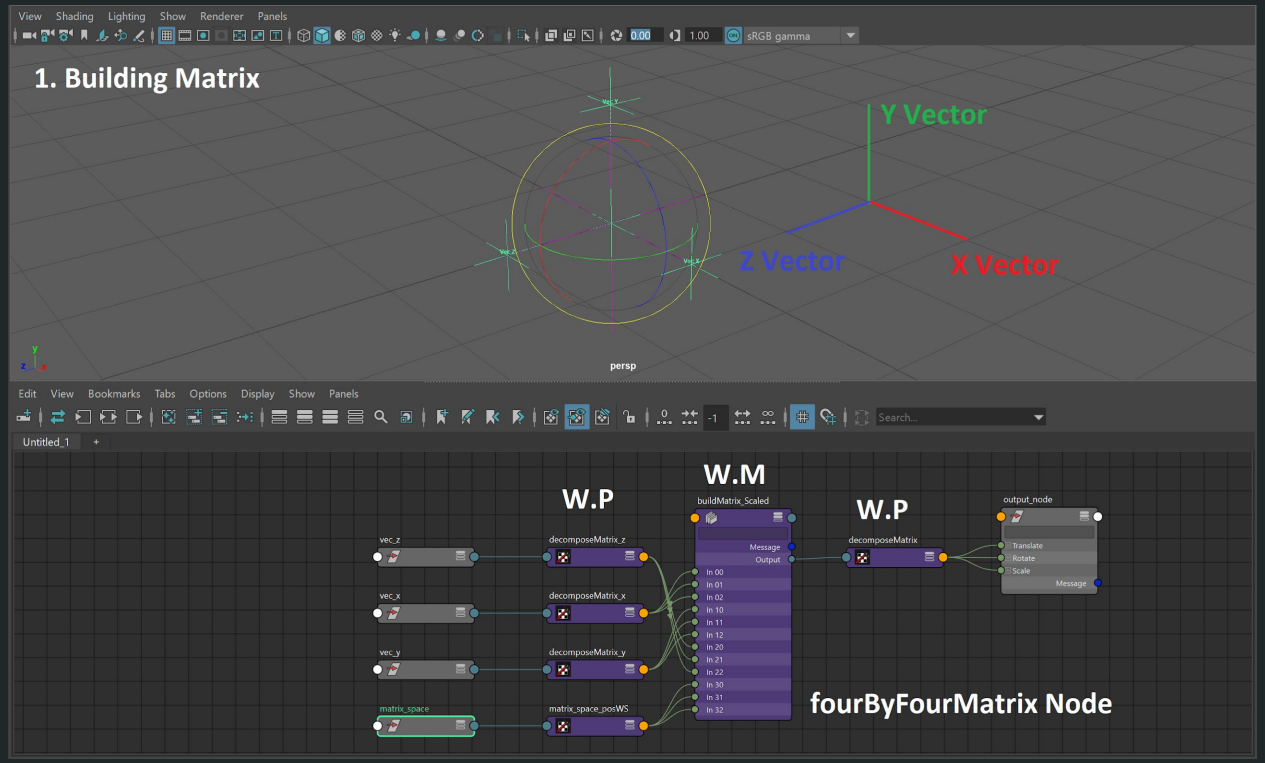
Rigging

- Vector



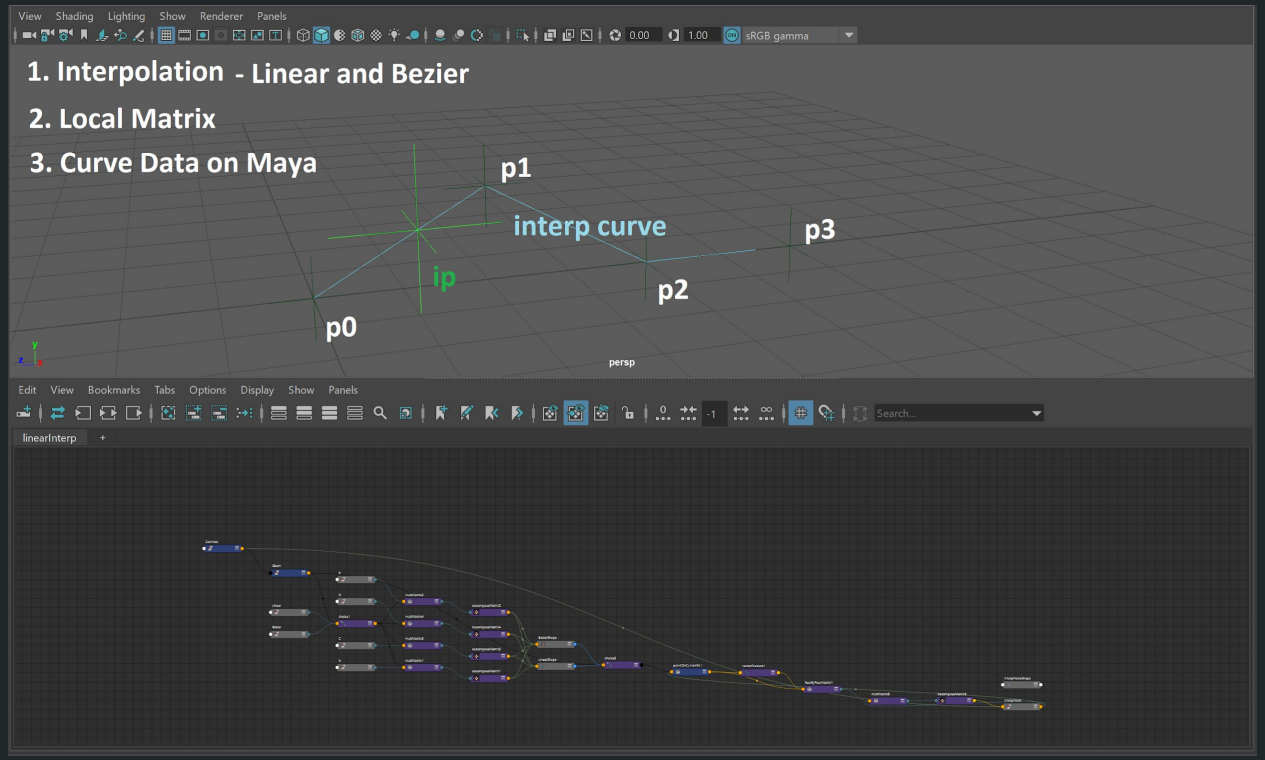
Rigging

- Matrix



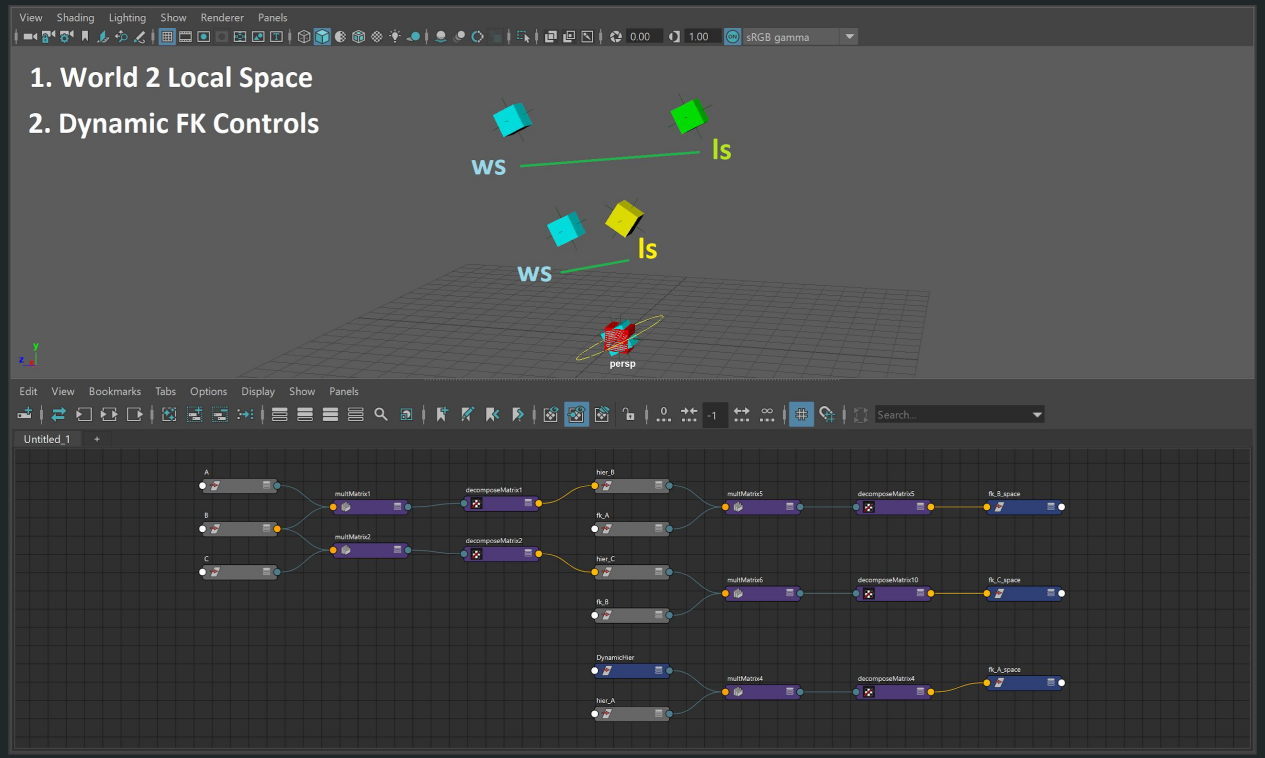
Rigging

- Interpolation



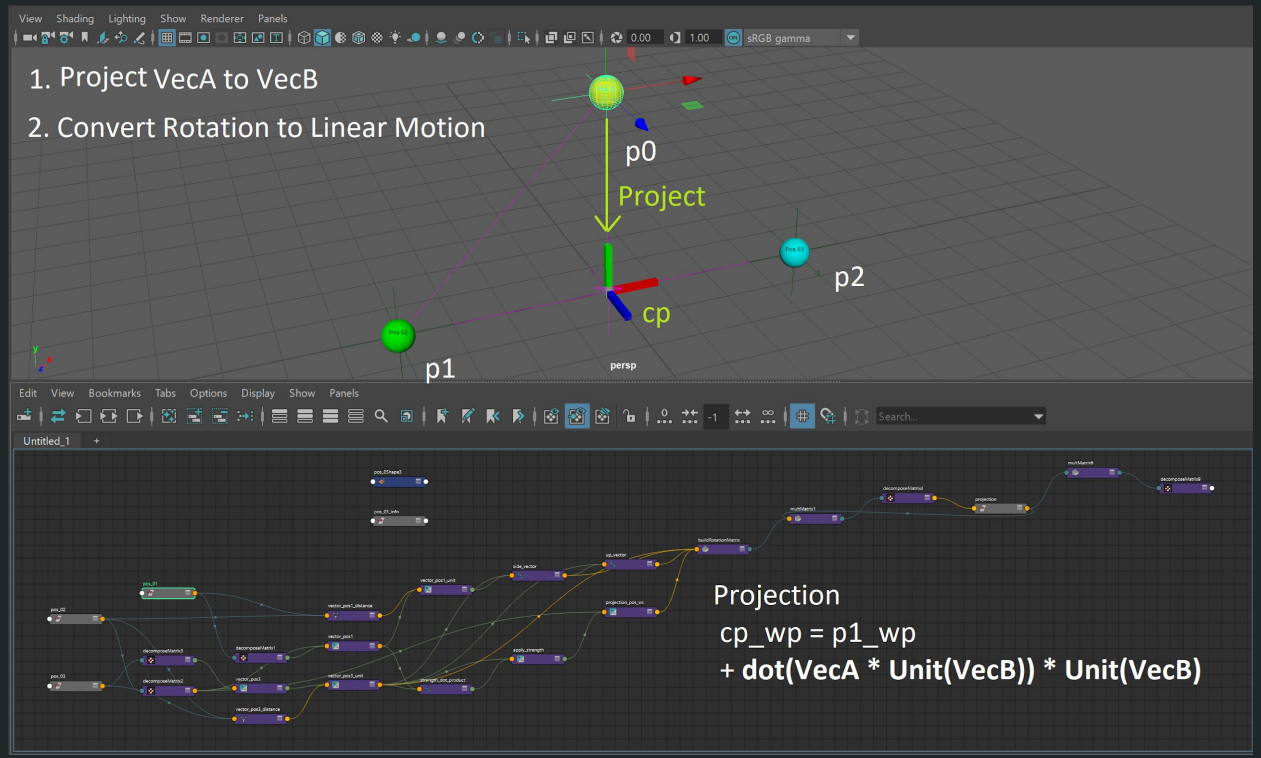
Rigging

- Space



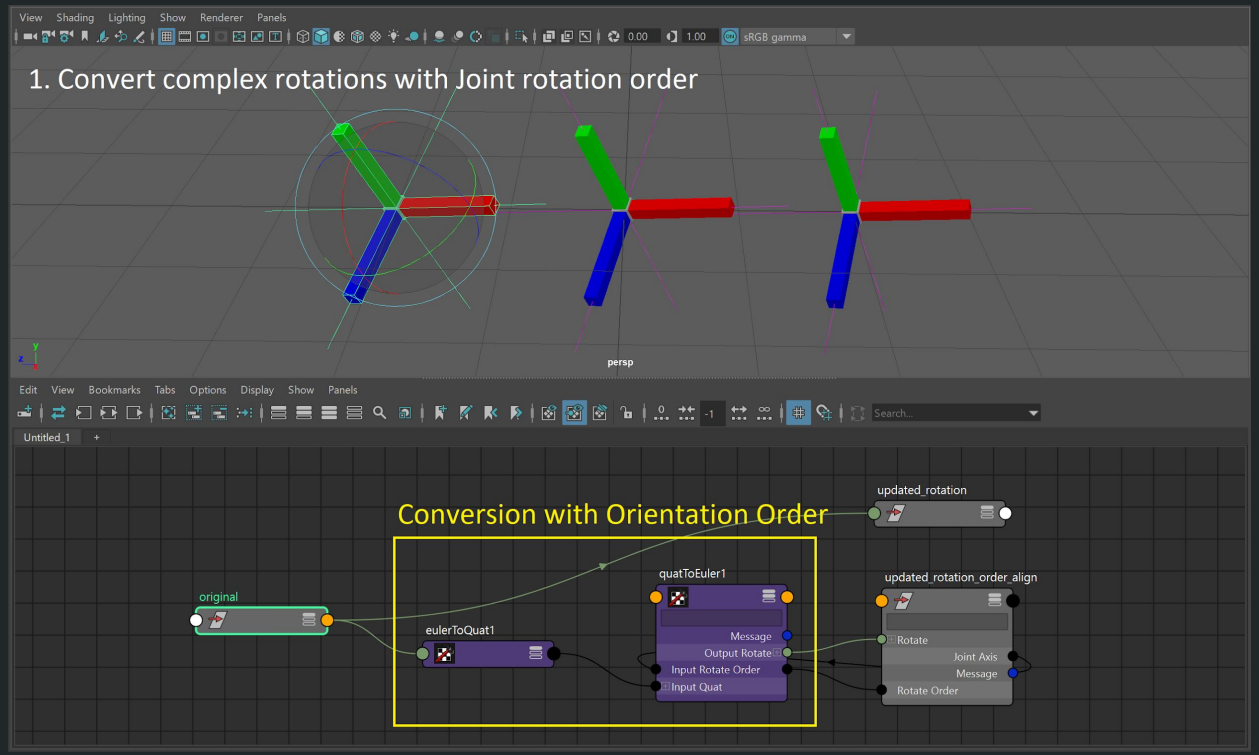
Rigging

- Project



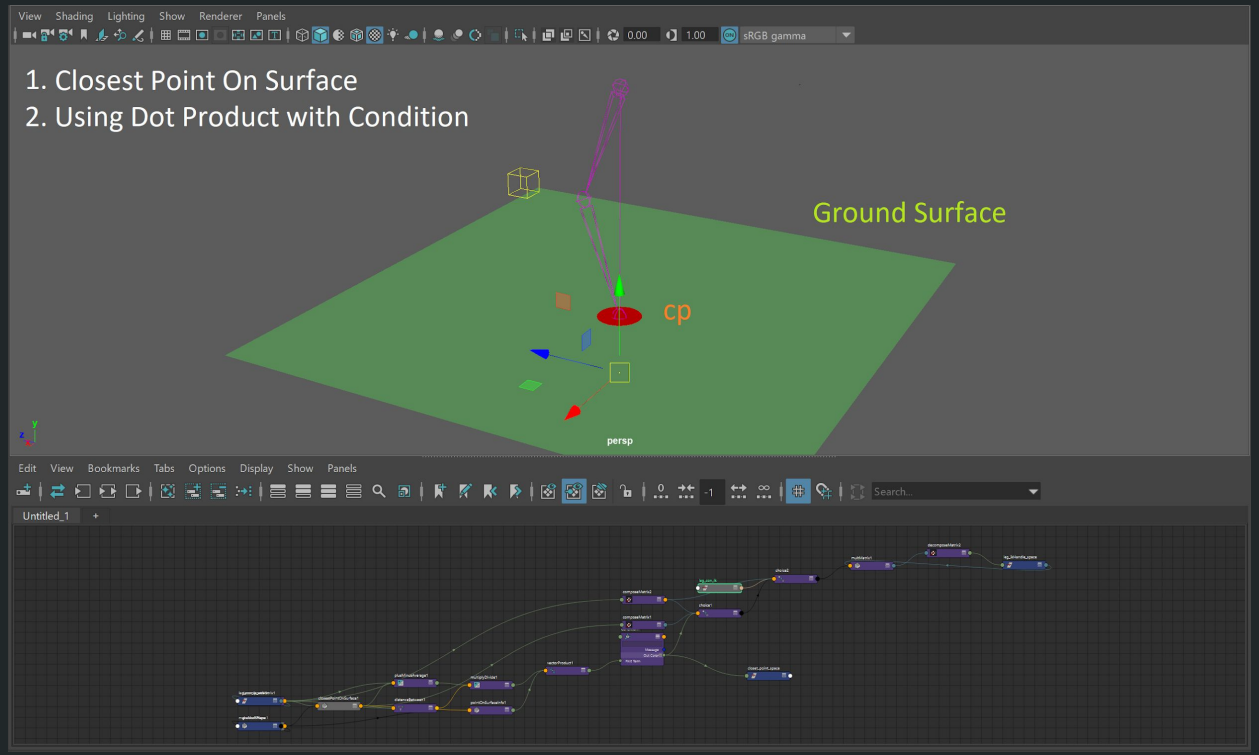
Rigging

- Axis Order



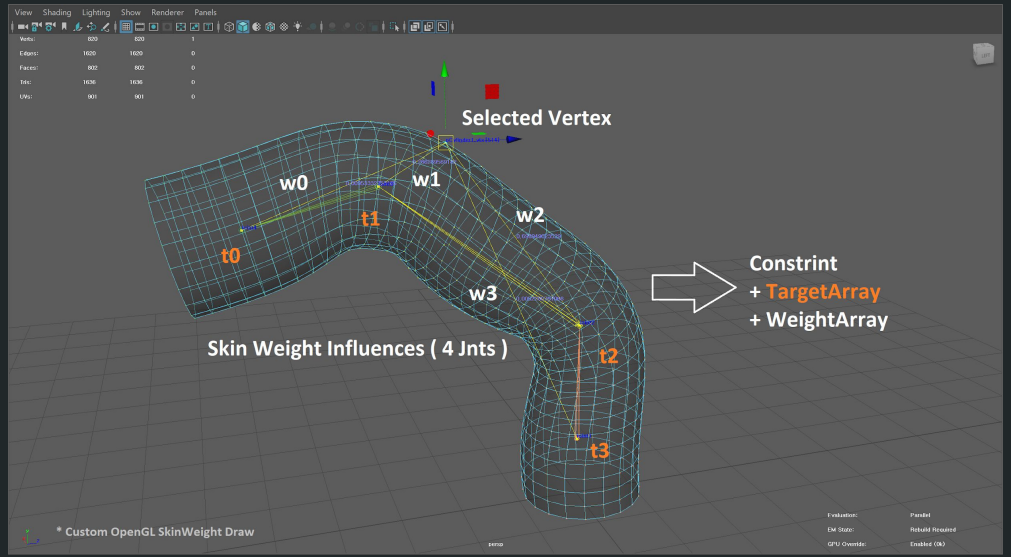
Rigging

- Ground Detect



Rigging

- Examples (Scripts)
  - Create Surface Constraint without using geometry operation
    i. Convert SkinWeights to Constraints



Rigging

- Access Data
  - Shaded git repository - “git clone git@github.com:artigee/Maya-Rigging-Example.git”
  - Bonus
    i. Script - convertSkinWeight2ParentConstraint.py
    ii. Plugins - curveLength2ParamU.py (for tail system)

Questions

- Not Limited to the today’s topics

Thank You

Contact

- Technical Art Director
- Hyun Seung Kim 김현승
- Email: artigee@gmail.com