Legend

High Level State	
Mid Level State	
Dispatcher	
Note to self	
Solution to Note	
Function	
Event	

Start → StakeTarget 1 Sequence

Choose Next StakeTarget

- Placed at StakeTarget2 skip to Hammering state
- Placed at StakeTarget3 skip to Hammering state

StakeTarget 2 Sequence ← stakesPlaced = 2 → StakeTarget 3 Sequence

- stakesPlaced = 3
- stakesPlaced = 3
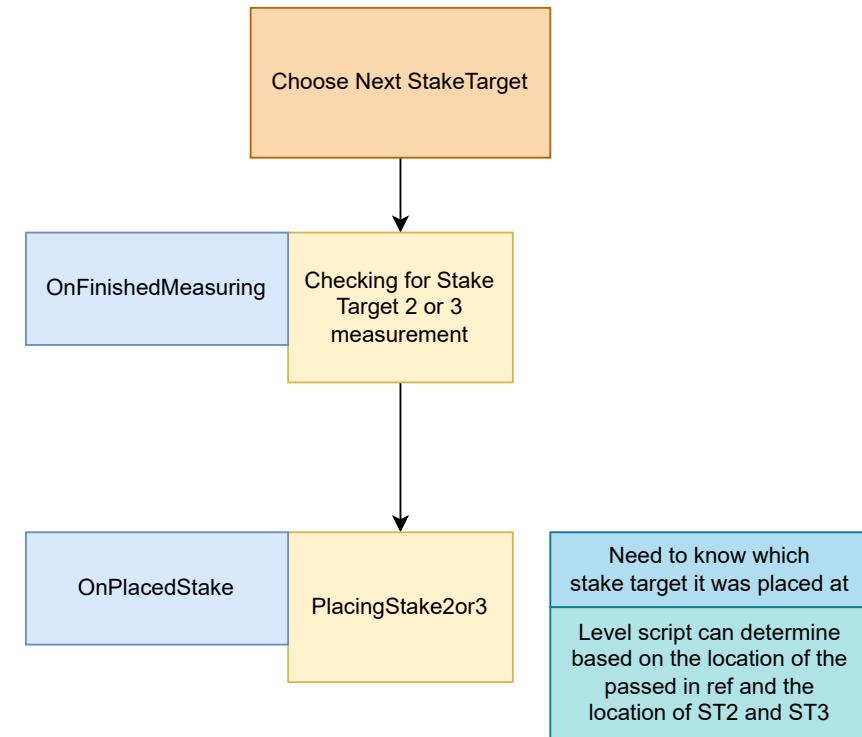
String Sequence

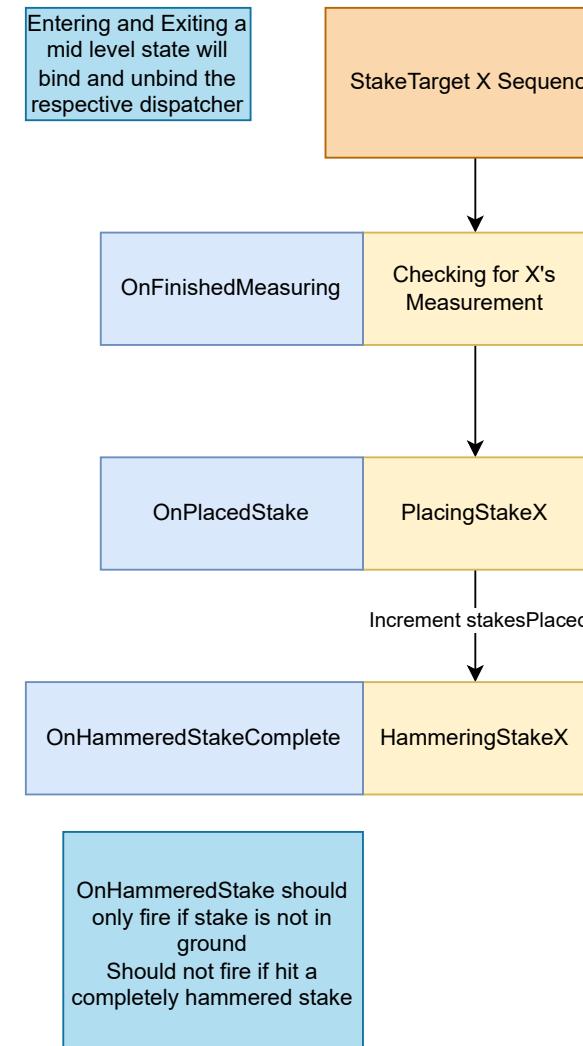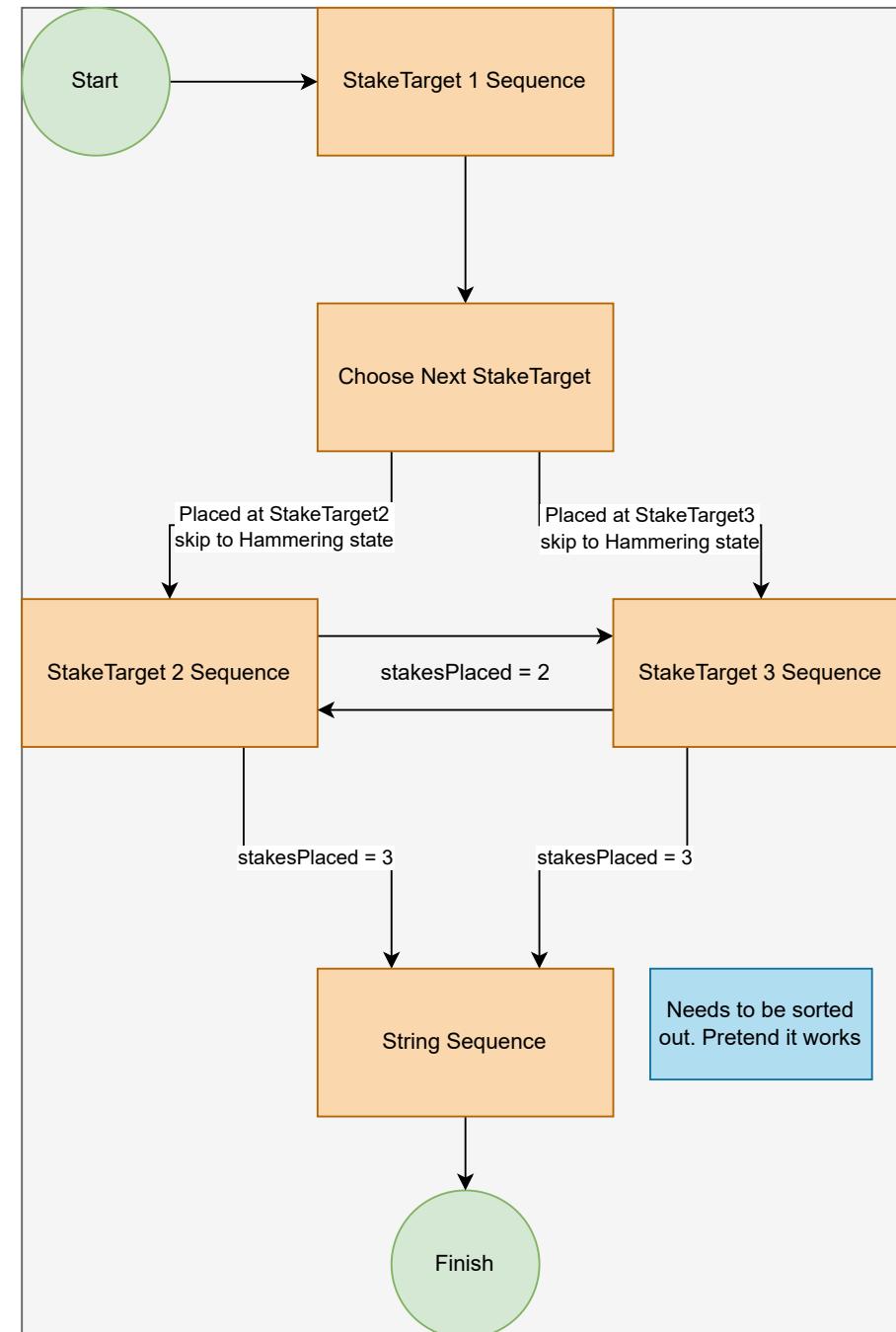Needs to be sorted out. Pretend it works
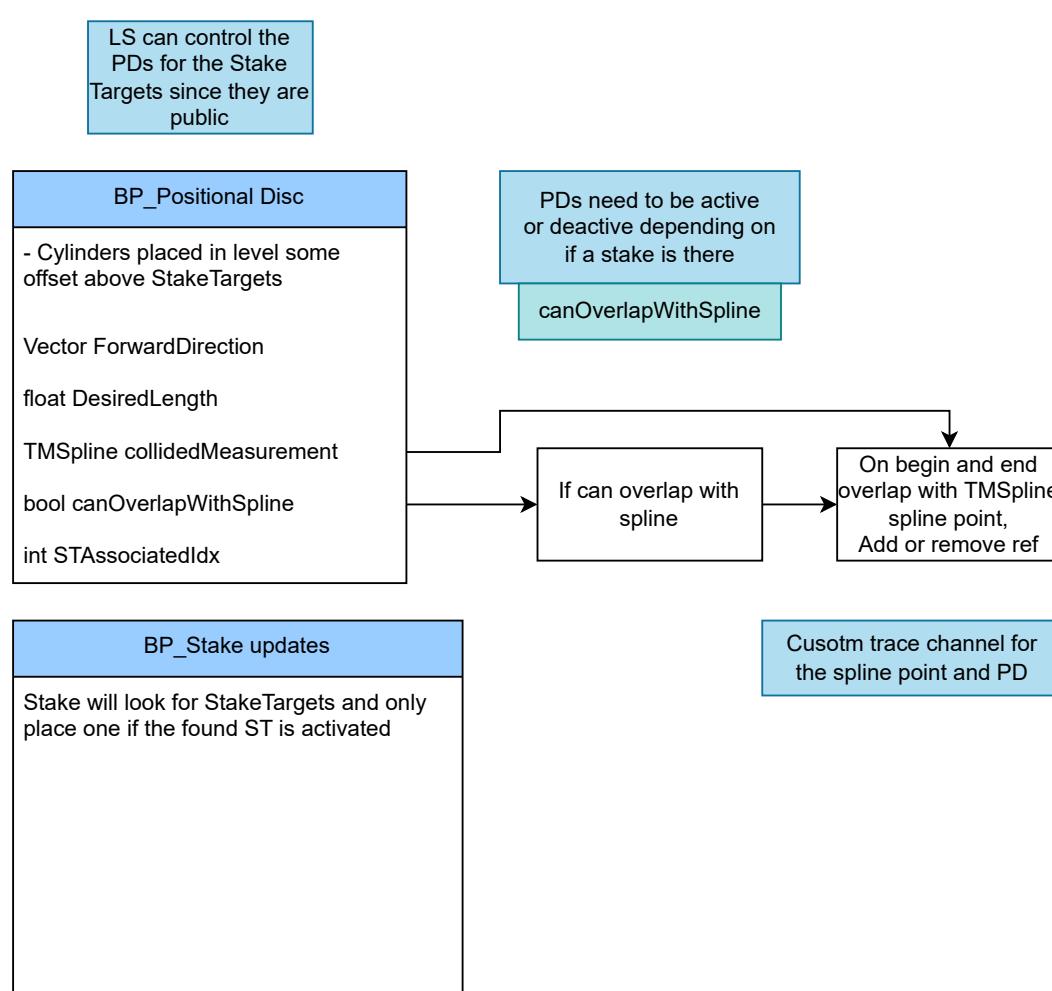
Finish

Lab4 vars

int stakesPlaced = 0
int stakesStrung = 0
float angleTol = 2
float lengthTol= 2
Array<StakeTarget> stakeTargetArray
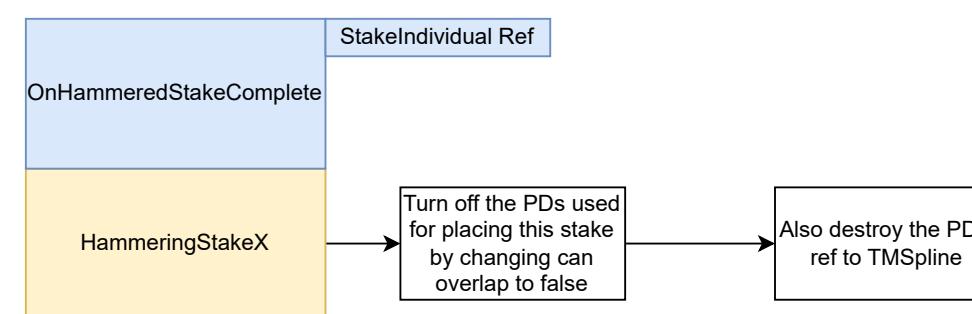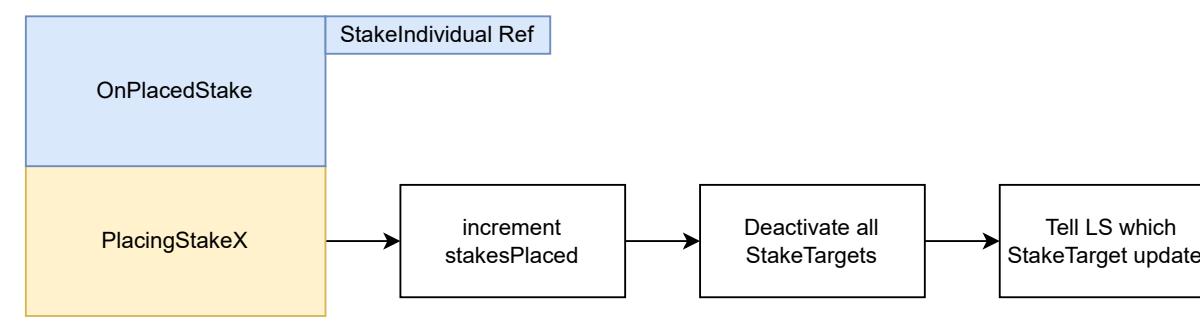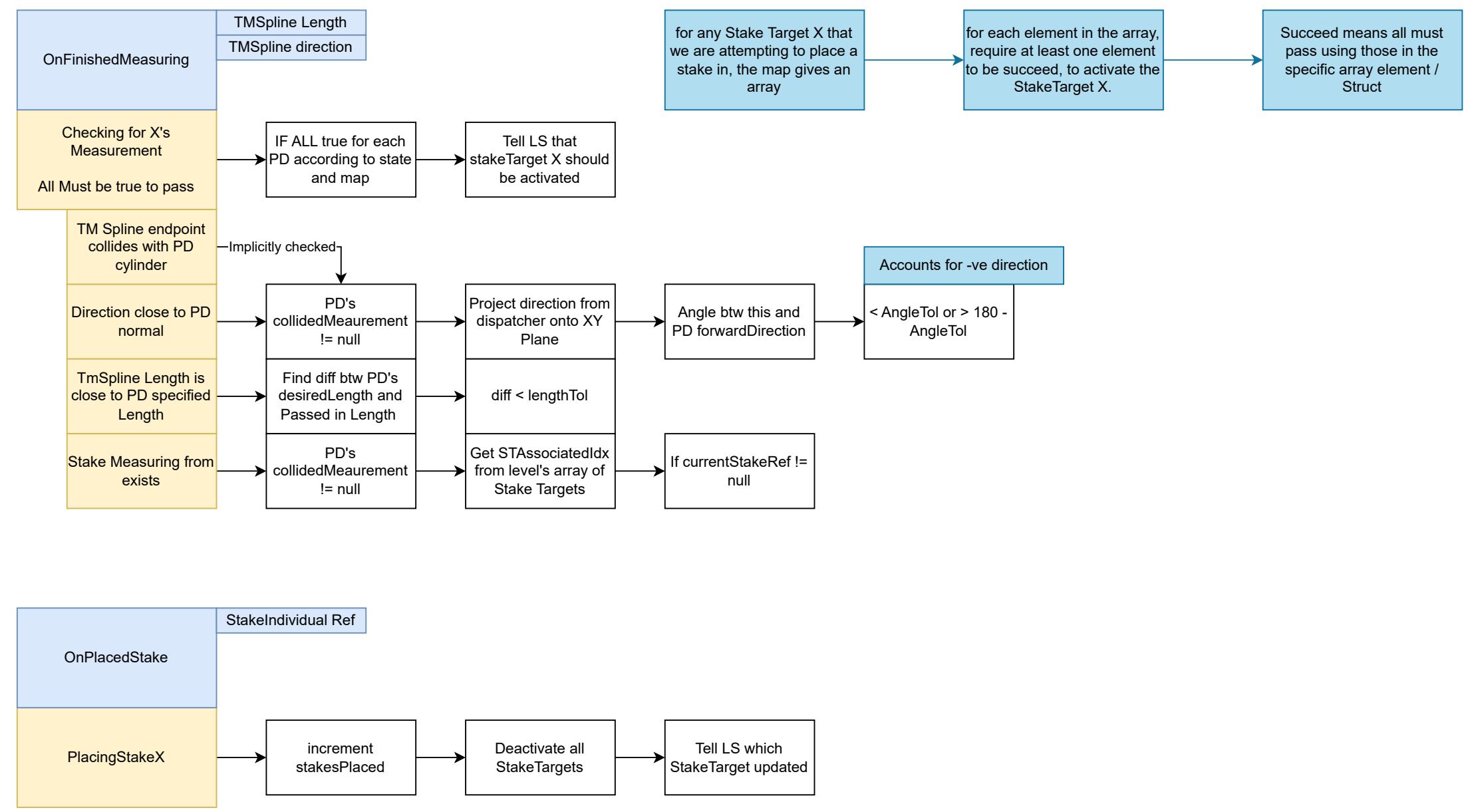Map<LevelState to Struct> MeasurementChecks

---

Entering and Exiting a mid level state will bind and unbind the respective dispatcher

StakeTarget X Sequence

OnFinishedMeasuring — Checking for X's Measurement

OnPlacedStake — PlacingStakeX

Increment stakesPlaced

OnHammeredStakeComplete — HammeringStakeX

OnHammeredStake should only fire if stake is not in ground
Should not fire if hit a completely hammered stake

Choose Next StakeTarget

OnFinishedMeasuring — Checking for Stake Target 2 or 3 measurement

OnPlacedStake — PlacingStake2or3

Need to know which stake target it was placed at

Level script can determine based on the location of the passed in ref and the location of ST2 and ST3

Old name: ValidStakePoints (VSP)

BP_ Stake Targets

bool isActivated
BP_StakeIndividual CurrentStakeRef

LS can control the PDs for the Stake Targets since they are public

BP_ Positional Disc

- Cylinders placed in level some offset above StakeTargets

Vector ForwardDirection
float DesiredLength
TMSpline collidedMeasurement
bool canOverlapWithSpline
int STAssociatedIdx

PDs need to be active or deactive depending on if a stake is there

canOverlapWithSpline

If can overlap with spline → On begin and end overlap with TMSpline spline point, Add or remove ref

TMSpline vars updates

Vector ForwardDirection

BP_Stake updates

Stake will look for StakeTargets and only place one if the found ST is activated

Cusotm trace channel for the spline point and PD

Enum -> Array<Struct>

MeasurementChecksMap: E_STAccess -> Array<Struct>

ST1	{PD(0, N)} ST1	
ST2	{PD(0,NE), PD(1, E)} ST2	{PD(0, NE), PD(2, S)} ST2
ST3	{PD(1, SE), PD(0, E)} ST3	{PD(1, SE), PD(2, S)} ST3

Struct for Map/Array

Array<BP_PositionalDisc> PDtoCheck
StakeTarget STToActivate

PDs are uniquely identified by the stake target they sit over and the direciton they face

---

OnFinishedMeasuring / TMSpline Length / TMSpline direction

Checking for X's Meaurement
All Must be true to pass

IF ALL true for each PD according to state and map → Tell LS that stakeTarget X should be activated

TM Spline endpoint collides with PD cylinder — Implicitly checked

Direction close to PD normal → PD's collidedMeaurement != null → Project direction from dispatcher onto XY Plane → Angle btw this and PD forwardDirection → < AngleTol or > 180 - AngleTol

Accounts for -ve direction

TmSpline Length is close to PD specified Length → Find diff btw PD's desiredLength and Passed in Length → diff < lengthTol

Stake Measuring from exists → PD's collidedMeaurement != null → Get STAssociatedIdx from level's array of Stake Targets → If currentStakeRef != null

for any Stake Target X that we are attempting to place a stake in, the map gives an array → for each element in the array, require at least one element to be succeed, to activate the StakeTarget X. → Succeed means all must pass using those in the specific array element / Struct

---

OnPlacedStake / StakeIndividual Ref

PlacingStakeX

increment stakesPlaced → Deactivate all StakeTargets → Tell LS which StakeTarget updated

---

OnHammeredStakeComplete / StakeIndividual Ref

HammeringStakeX

Turn off the PDs used for placing this stake by changing can overlap to false → Also destroy the PD's ref to TMSpline