Plan such that a curve of interest/drama/tension is maintained, while each piece of information which requires grounding is grounded.

NarrativeEvent class

Has:

What information we want to present. Several types:

Shape Part (whole shape, left side, right side, first leg, etc.)

Has list of critical points involved in the shape part (list of critical points)

Has name of shape part.

Has normal values for shape part

How tall it’s supposed to be (y-difference)

How wide it’s supposed to be (x-difference)

Critical point

Has the point itself (x/y) ^

Has the normal value for the point (x/y) ^

Abnormality

Has shape part involved in it

Has the critical points involved in it (list of critical points) ^

Has how abnormal it is (float) ^

Grounding requirement ^

Shape parts have a critical point grounding requirement.

Abnormalities have a critical point grounding and/or shape part grounding requirement.

Tension amount

If it’s an abnormality, this is based on how abnormal it is.

GenerateNarrative

Order events:

First, define a function for the narrative curve.

Next, using the turn limit, select events such that the narrative curve is satisfied.

Story shaping variables:

Pre-grounding weight: how much we value grounding before something has been mentioned.

Post-grounding weight: how much we value grounding after something has been mentioned.

Tension curve: The optimal tension for the story, in terms of float values per narrative event. (e.g. at narrative event 0, we want x tension). Should scale to any number of narrative events.

Story size: how many narrative events the story should be.

Relevant/changed classes:

Program.cs (changed how it was called so it can be used standalone)

NarrativeEvent.cs (added new variables, change functionality of narrative event)

New classes:

GraphInfo

Abnormality

CriticalPoint