The following criteria are listed in order of importance starting with the most important.

Computations easy.

Description: A rating on how easy the computations are for the conversion of GRUE-42 to matlab. This is the most important criteria as most of the complexity lies within the computations.

Rank: 70. Every rule in the AST was built with the computations in mind. Since almost every aspect of our tree requires a computation, rather than try to reduce the number of computations and rules needed, we simply associated a rule with almost every node. While it makes the LIDO file somewhat larger, it should greatly simplify the computations needed later on. We're sure there are probably ways which we could simplify the computations even further, we just don't know what they are, hence the rank of 70 (with 100 being the most simple computations possible).

Source Program User Friendly

Description: Aerospace majors hate coding. We want to make the use of GRUE-42 as painless an intuitive as possible, so that the program will be worth using, and our friends won't hate us for making a difficult to use language if they ever try it.

Rank: 85. The input language is very simple. A minimal number of input parameters are required to correctly generate an orbit, and the format is similar to matlab, so it should be pretty intuitive for the user if they've used matlab before, and should be very easy to learn even if they haven't. There's probably some room for improvement here, but the complexity that any improvement would add would probably not be worth the increase in ease of use.

Easy to change.

Description: There is a very good chance that the language will not be correct the first time, thus being able to easily change it will be useful. We would also like to continue to expand this language and add complexity, which will only be worth doing if the language is decently easy to change. Otherwise the time needed to change the language would be more than the time required to just do the Matlab code ourselves.

Rank: 65. The language is laid out in such a way that it is fairly easy to expand. If you want to add a new type of orbital computation, you just add a new rule for it under the CompList. Since all of the orbital parameters will be stored as properties, new computations shouldn't have any trouble getting the information they require.

Readable.

Description: Applies to the readability of all the language files and how easy they are to understand. We want the files to be fairly self explanatory, and not wonder what the hell we were doing when we go back to try to modify them.

Rank: 70. We tried to name everything in a way that shows the structure of the tree, like the Computations section has a CompList, which is just a list of comps. We also named things to reflect the values they're holding, for example, the declaration of the semi-major axis is SemiMajorAxisDecl. The sheer number of rules lowers readability slightly.

Min. number of rules.

Description: This ranking is designed to help reduce the total number of rules that are in the AST. The less rules the less complex the grammar becomes. We also have to write less.

Rank: 50. We have a lot of rules because we're trying to make the computations as easy as possible, but we don't
have unnessecary rules. We also have some added complexity because we wanted keywords to make it more user friendly, like "Satellite" and "End". For the parts that don't require computations, they were put in the con file so that we didn't have to have a separate rule for every possibility.

Determines order of execution.

Description: Some of the calculations require other calculations to be done first. We have to ensure that those calculations take place in the nessecary order. We also don't want things to be put in the ptg nodes before they have anything in them.

Rank: ?. We don't forsee any problems in this area, but since we haven't done the computations yet we may yet run into unexpected issues in this area. If this is the case, the tree may need to be modified.

ATTR Value, Alt, Ecen : float;

RULE: Program LISTOF Object END;

/* Satellite Info */
RULE: SatelliteDef::= 'Satellite' NameDef SatelliteDesc 'End' COMPUTE
END;

RULE: SatelliteDesc LISTOF SatParam END;

RULE: SatParam ::= OrbitType COMPUTE
END;
RULE: SatParam ::= SemiMajorAxisDecl COMPUTE
END;
RULE: SatParam ::= SemiMinorAxisDecl COMPUTE
END;
RULE: SatParam ::= LinEccentDecl COMPUTE
END;
RULE: SatParam ::= EccentricityDecl COMPUTE
END;
RULE: SatParam ::= CenterDesc COMPUTE
END;
RULE: SatParam ::= MassDesc COMPUTE
END;
RULE: SatParam ::= EpochDesc COMPUTE
END;
RULE: SatParam ::= LifetimeDecl COMPUTE
END;
RULE: SatParam ::= InclDecl COMPUTE
RULE: OrbitType ::= 'Orbit' '=' 'LEO' COMPUTE
    OrbitType.Alt = 0;
    OrbitType.Ecen = 0;
END;

RULE: OrbitType ::= 'Orbit' '=' 'GEO' COMPUTE
    OrbitType.Alt = 0;
    OrbitType.Ecen = 0;
END;

/* Planet Rules */
RULE: PlanetDef ::= 'Planet' NameDef PlanetDesc 'End' COMPUTE
    /* Use PlanetDesc.Mass in your PTG computations */
    /* Name Analysis done here */
END;

RULE: PlanetDesc LISTOF PlanetParam END;

RULE: PlanetParam ::= PlanetType COMPUTE
END;
RULE: PlanetParam ::= SemiMajorAxisDecl COMPUTE
END;
RULE: PlanetParam ::= SemiMinorAxisDecl COMPUTE
END;
RULE: PlanetParam ::= LinEccentDecl COMPUTE
END;
RULE: PlanetParam ::= EccentricityDecl COMPUTE
END;
RULE: PlanetParam ::= CenterDesc COMPUTE
END;
RULE: PlanetParam ::= MassDesc COMPUTE
END;
RULE: PlanetParam ::= TiltDecl COMPUTE
END;

RULE: PlanetType ::= 'Earth' COMPUTE
END;
RULE: PlanetType ::= 'Moon' COMPUTE
END;
RULE: PlanetType ::= 'Sun' COMPUTE
END;
/* Orbit Parameter Info*/

RULE: EpochDesc::= 'Epoch' '=' Number COMPUTE
    EpochDesc.Value = Number;
END;

RULE: CenterDesc::= 'Center' '=' PlanetType COMPUTE
END;

RULE: CenterDesc::= 'Center' '=' Coordinate COMPUTE
END;

RULE: MassDesc::= 'Mass' '=' Number COMPUTE
END;

RULE: SemiMajorAxisDecl::= 'a' '=' Number COMPUTE
END;

RULE: SemiMinorAxisDecl::= 'b' '=' Number COMPUTE
END;

RULE: LinEccentDecl::= 'c' '=' Number COMPUTE
END;

RULE: EccentricityDecl::= 'e' '=' Number COMPUTE
END;

RULE: Coordinate::= '('Number','Number','Number')' COMPUTE
END;

RULE: LifetimeDecl::= 'Lifetime' '=' Number COMPUTE
END;

RULE: InclDecl::= 'I' '=' Number COMPUTE
END;

RULE: TiltDecl::= 'Tilt' '=' Number COMPUTE
END;

/* Computation Rules */

RULE: Computations ::= 'Computations' CompList 'End' END;
RULE: CompList LISTOF Comp END;

RULE: VaReq::= 'Va' '(' NameUse ')' COMPUTE
RULE: VpReq::= 'Vp' '(' NameUse ')' COMPUTE END;
RULE: PReq::= 'P' '(' NameUse ')' COMPUTE END;
RULE: ShadowReq::= 'Shadow' '(' NameUse ')' COMPUTE END;
RULE: PositionReq::= 'Position' '(' NameUse ')' COMPUTE END;
RULE: DistanceReq::= 'Distance' '(' DistanceObject ',' DistanceObject ')' COMPUTE END;
RULE: DistanceObject ::= PlanetType END;
RULE: DistanceObject ::= NameUse END;

/* Saving the Terminal Name in a Sym attribute so
we can do NameAnalysis on the Name! */
SYMBOL NameDef COMPUTE
SYNT.Sym = TERM;
END;

SYMBOL NameUse COMPUTE
SYNT.Sym = TERM;
END;
&]

Object: SatelliteDef / PlanetDef / Computations.
NameDef: Identifier.
Comp: VaReq / VpReq / PReq / ShadowReq / PositionReq / DistanceReq.
NameUse: Identifier.
[
]
@O@<GRUE-42.map@>==[@- MAPSYM
MAPRULE
@]