Juan Sedano
Introduction: Realms of Hyrule Character Generator

-Background information:

Originally the system was an entirely manual process of character creation ala D&D. And this process was a little bit more strenuous than any given table top system before it. Their table top system was a customization of a D&D spin-off and thus everyone who wanted to partake needed to go through a process of understanding how to create a character under this new system which they created, even if it did undergo strenuous testing, new always has a learning curve. So in order to speed up the process i began creating a generator for them around august of this year that would streamline the process.
-The mechanics:
The core of the system is that basic triggers and math is at the core of the idea with all of the myriad of rules built into the system. Basic functionality included a series of checks and balances and situational triggers. Any attempt at explaining the rules and system in a succinct manner will prove inadequate.

-Version One:
http://realmsofhyrule.net/character/roh_chargen_publicbeta.php
Essentially what happens here is a highly procedural and early version of the code. It works and has most of the rules done, but because it's procedural and all of it works on the firing of certain procedures and functions on clicks, it's less than ideal. All the information is also stored in the javascript file itself and while works, it makes the code less than ideal. The procedurally of it actually causes bugs between certain functions because since they all fire regardless of external information when called, the information can stack up and bug out. The procedural nature of it also meant that updating the core information upon the leftmost two columns changing was near impossible without slowing down the code significantly. Thus in order to take into account the changes it was necessary to tie it to a third, separate, set of functions that would update and reset information because there was no way to keep track of old information efficiently. This created an in-between because of the necessity of procedurally. Thus the "Update Race/Archetype" button was created, a less than ideal solution to

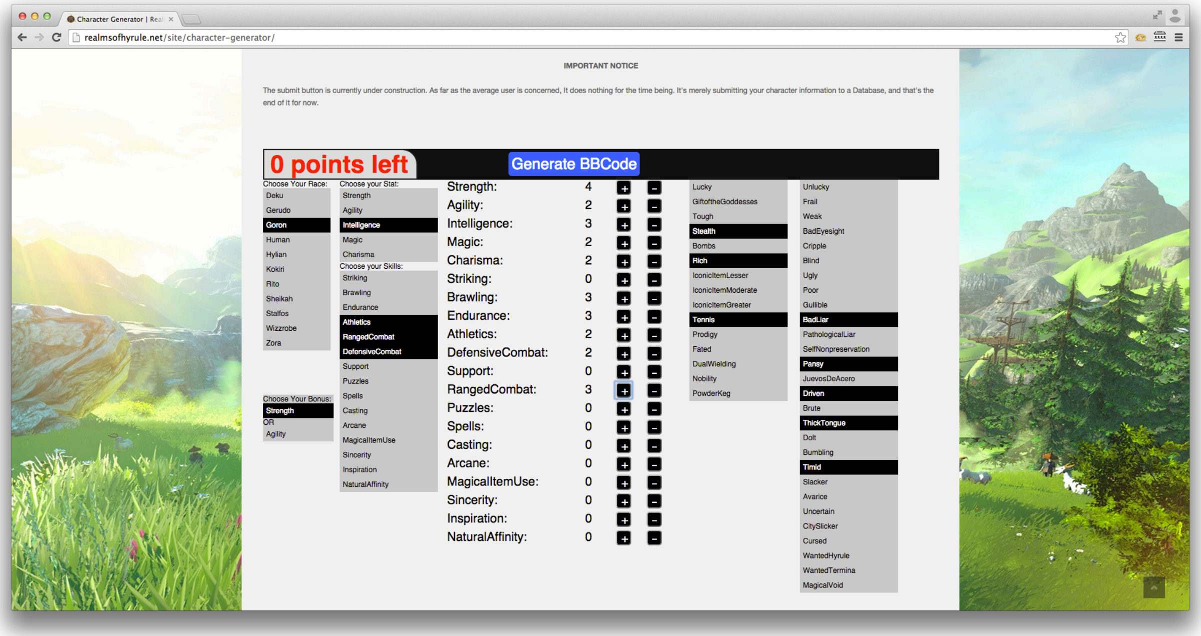
the problem.

-Version Two:
Link TBD
So this version saw the core functionality of the code get increasingly more efficient. The code went from being highly procedural to Object-Oriented. The information is no longer stored in arrays on the javascript file but in databases which allow for quick manipulation of the front end without much hassle. No longer is the code based on procedures and functions but on classes and methods which make everything operate off a central set of properties which allow for a smooth transition between information because it's all relevant and comes from a source object. The information coming from the database and being fed into classes created robust and functional code that no longer needed any delay and worked just fine regardless of most other things. The code is elegant and legible and was created in such a way that it's easy to manipulate. Clunky functions within functions and long and tiresome if/else strings were replaced with succinct methods and switches or a single for each or for in or some variation thereof. The code was made clean and elegant and the core of it began to be commented out. At this stage the core idea of a generator was complete, with most aspects of it being automated through methods and information called from database information, with little manual hard coding. This version saw the implementation of RESTful applications and MVC (Model, View, Controller) to create a streamlined, fast, efficient, and responsive. The Model takes place of the Javascript object ActiveCharacter and it updates the view, which the user interacts with. Unlike in previous versions where there was no separation between what the user saw and what the numbers reflected, now the view is simply supposed to reflect what the model contains. Submition and creation depends on the client aka the DB.

-Version three beta:
http://www.jsed.net/other/chargen/index.php
Once the core code was complete, the goal then moved from a basic one time generator to a repeatable, re-usable and functional generator and tracker. Beyond the initial creation, the goal was now to make the user be able to further the information and be able to submit these initial sheets to a database for storage and later re-using. Because the sheets are an ever evolving thing where characters need to be able to change, and even after

the original generation manual input was still necessary, so the next step was to allow for that process to be automated too. Currently this is being tested in beta and is following the same mantra of the previous iteration: RESTful API with MVC implementations. Code for this version can be found below. When using this version, selecting "continue as guest" will lead you to the version mentioned in version two, only the JS file, which still isn't correctly split, has extraneous functions and methods that are used for logged characters. This version also has experimentations with advanced security measures and PHP SESSION and cookie handling to keep track of these.

Code:
http://www.jsed.net/other/chargen/chargen.zip


Character Generator | Real
realmsofhyrule.net/site/character-generator/

IMPORTANT NOTICE

The submit button is currently under construction. As far as the average user is concerned, It does nothing for the time being. It's merely submitting your character information to a Database, and that's the end of it for now.

0 points left

Choose Your Race: Deku, Gerudo, **Goron**, Human, Hylian, Kokiri, Rito, Sheikah, Stalfos, Wizzrobe, Zora

Choose Your Bonus: **Strength** OR Agility

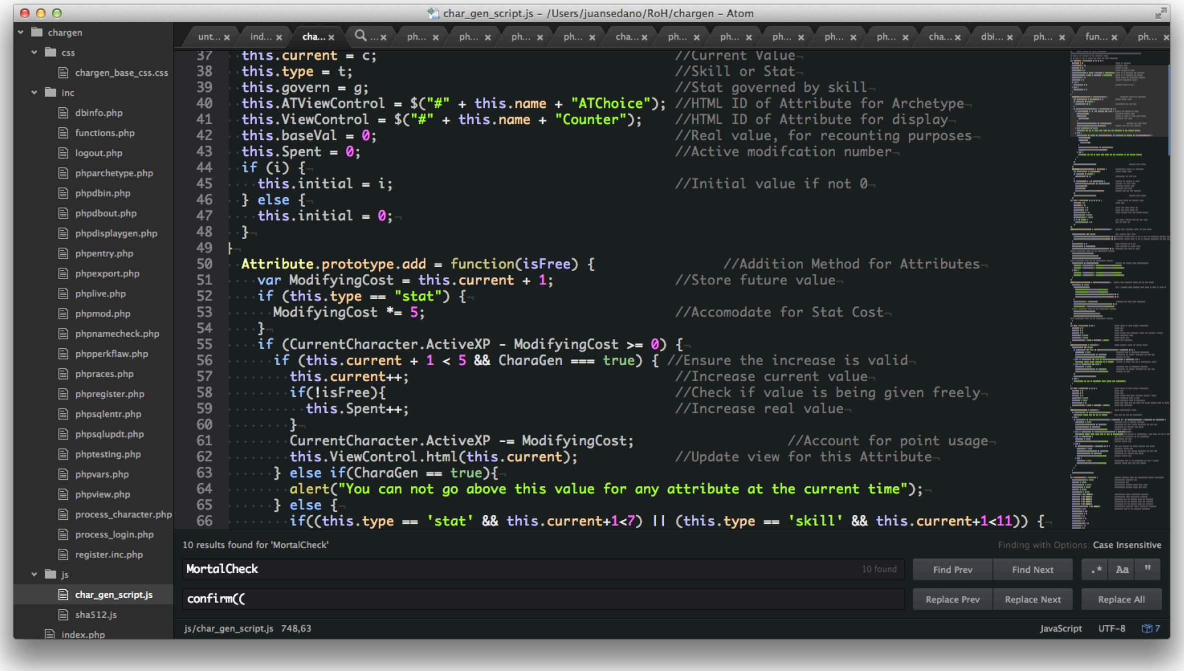
Choose your Stat: Strength, Agility, **Intelligence**, Magic, Charisma

Choose your Skills: Striking, Brawling, Endurance, **Athletics**, **RangedCombat**, **DefensiveCombat**, Support, Puzzles, Spells, Casting, Arcane, MagicalItemUse, Sincerity, Inspiration, NaturalAffinity

Strength:	4	+	−
Agility:	2	+	−
Intelligence:	3	+	−
Magic:	2	+	−
Charisma:	2	+	−
Striking:	0	+	−
Brawling:	3	+	−
Endurance:	3	+	−
Athletics:	2	+	−
DefensiveCombat:	2	+	−
Support:	0	+	−
RangedCombat:	3	+	−
Puzzles:	0	+	−
Spells:	0	+	−
Casting:	0	+	−
Arcane:	0	+	−
MagicalItemUse:	0	+	−
Sincerity:	0	+	−
Inspiration:	0	+	−
NaturalAffinity:	0	+	−

Lucky, GiftoftheGoddesses, Tough, **Stealth**, Bombs, **Rich**, IconicItemLesser, IconicItemModerate, IconicItemGreater, **Tennis**, Prodigy, Fated, DualWielding, Nobility, PowderKeg

Unlucky, Frail, Weak, BadEyesight, Cripple, Blind, Ugly, Poor, Gullible, **BadLiar**, PathologicalLiar, SelfNonpreservation, **Pansy**, JuevosDeAcero, **Driven**, Brute, **ThickTongue**, Dolt, Bumbling, **Timid**, Slacker, Avarice, Uncertain, CitySlicker, Cursed, WantedHyrule, WantedTermina, MagicalVoid



```
  this.current = c;                                           //Current Value
  this.type = t;                                              //Skill or Stat
  this.govern = g;                                            //Stat governed by skill
  this.ATViewControl = $("#" + this.name + "ATChoice"); //HTML ID of Attribute for Archetype
  this.ViewControl = $("#" + this.name + "Counter");    //HTML ID of Attribute for display
  this.baseVal = 0;                                           //Real value, for recounting purposes
  this.Spent = 0;                                             //Active modifcation number
  if (i) {
    this.initial = i;                                         //Initial value if not 0
  } else {
    this.initial = 0;
  }
}
  Attribute.prototype.add = function(isFree) {                    //Addition Method for Attributes
    var ModifyingCost = this.current + 1;                     //Store future value
    if (this.type == "stat") {
      ModifyingCost *= 5;                                     //Accomodate for Stat Cost
    }
    if (CurrentCharacter.ActiveXP - ModifyingCost >= 0) {
      if (this.current + 1 < 5 && CharaGen === true) { //Ensure the increase is valid
        this.current++;                                       //Increase current value
        if(!isFree){                                          //Check if value is being given freely
          this.Spent++;                                       //Increase real value
        }
        CurrentCharacter.ActiveXP -= ModifyingCost;                   //Account for point usage
        this.ViewControl.html(this.current);                  //Update view for this Attribute
      } else if(CharaGen == true){
        alert("You can not go above this value for any attribute at the current time");
      } else {
        if((this.type == 'stat' && this.current+1<7) || (this.type == 'skill' && this.current+1<11)) {
```

10 results found for 'MortalCheck'

MortalCheck

confirm((

js/char_gen_script.js 748,63