**ADVENTURER**

A Fantasy Role-Playing Adventure Program

for the Chronos Gaming Framework

Developers’ Reference Guide

by

Alan Cline

February 12, 2002

Rev. Feb 5, 2011

**Table of Contents**

1. Introduction 4

1.1. Adventurer Description 4

1.2. Creating a New Character 6

1.3. General Design Layout 6

1.4. Architecture and Software Architecture Driving Principles 6

1.5. Attribute Types and Issues 7

1.5.1. Attribute Types 7

1.5.2. Issue 8

1.5.3. Recommendation 8

Example: Person’s Speed 8

1.6. Future Versions 9

1.7. Related Documentation 9

2. Collect the Player’s Input 10

3. Generate the Hero 12

3.1. Save the Player Input 12

3.2. Set the Initial Age 12

3.3. Set the Prime Traits for Peasant 12

3.4. Adjust Traits for Gender 13

3.5. Adjust Traits for Age 13

3.6. Adjust Traits for Race. 13

3.7. Verify the Race Limits 14

3.8. Find the Height and Weight 14

3.9. Calculate the Modifiers for Each Prime Trait 15

3.9.1. Assign Strength Modifiers 15

3.9.2. Set Languages from Intelligence 16

3.9.3. Assign Wisdom Modifiers 17

3.9.4. Assign Constitution Modifiers 17

3.9.5. Assign Dexterity Modifiers 18

3.9.6. Assign Charisma Modifiers 18

3.10. Initialize Derived Attributes 18

3.10.1. Calculate Initial Hunger State 18

3.10.2. Set Level and Experience Points 19

3.10.3. Set Hit Points 19

3.10.4. Set Initial Gold 19

3.11. Set Starting Inventory and Related Attributes 20

3.11.1. Calculate Action Points and Modifiers 21

3.11.2. Set Armor Class (AC) 21

3.11.3. Set Base Movement 22

3.12. Add Skills and Special Abilities 22

3.12.1. Assign Literacy Skills 22

3.12.2. Assign Skills from Occupations 23

3.12.3. Assign Racial Skills 23

Dwarf Abilities 23

Elf Abilities 23

Gnome Abilities 23

Half-Elf Abilities 24

Hobbit Abilities 24

Half-Orc Abilities 24

4. Display the Hero’s Profile 25

4.1. Player’s Name Plate 25

4.2. Displaying the Hero Attributes 25

4.2.1. Row 1: XP, Level, HP, Occupation, Hunger 25

4.2.2. Row 2: AC, Speed, Age, Height, Weight 25

4.2.3. Row 3: Gold and Weight Carried 26

4.2.4. Row 4: Languages 26

4.2.5. Row 5: Hero’s Description 26

Putting these fragments together creates a somewhat varied description of the person that automatically reflects that Hero’s Charisma and body type. 27

4.3. Displaying the Hero’s Inventory 27

4.4. Displaying the Hero’s Skills and Special Abilities 27

# Introduction

*Adventurer* 1.0 is an adventure role-playing game for one person on a Mac OS workstation. It is part of the ***Chronos*** role-playing game framework. ***Chronos*** is comprised of two programs: an adventure authoring kit and editor, *Dungeon Wizard*, and its corresponding player module, *Adventurer*.

*Dungeon Wizard* enables a user to build a multiplayer game, or interactive novel; and *Adventurer* allows users to play the game that the author built. The D*ungeon Wizard* user is the adventure author who defines items, situations, obstacles, interactive player Character abilities, and non-player characters (NPC) to provide support or conflict that the *Adventurer* player must overcome. The author also defines the traits and kinds of "monsters", creatures or characters that the player's avatar meets in the environment, which is generically referred to as "the dungeon."

## Adventurer Description

*Adventurer* is for the general user and should be as easy to play as possible, using the *Dungeon Wizard* author's plots, dungeons, and playing tools. *Adventurer* has built-in commands the player gives to his or her Hero to explore, search, cast spells, get and drop items, and gain ability. The Hero moves about in a semi-text, semi-graphic environment; *Adventurer* displays the descriptions and reactions to the players' commands.

The initial *Adventurer* game comes with a set of default monsters, NPCs, skills, items, and other basics to allow a game to be played "out of the box" using a medieval fantasy genre. These fundamentals are generated by *Dungeon Wizard*, and can be customized by the game author. See the *Dungeon Wizard* Developers’ Guide and Player’s Manual for more detail on that game.

Currently, Chronos is a single-player game that allows the user to control up to four Heroes in the same game simultaneously. The player starts the Heroes by taking turns, but as different actions take different amounts of time to complete, and monsters and events pre-empt or interleave their actions, the game has a real-time "feel" to it, and each Hero takes its action as it can, one action following another.

The player’s Character, or avatar, is known as the Hero. *Adventurer* has the following corresponding modes.

* **Admin:** Create Characters (avatars), buy and sell inventory, and get promoted to higher levels of ability in various Guilds. A player's avatar starts in earnest after reaching 200 XP and becoming a member of a Guild. Admin modes also include saving Characters and collecting up to four Characters to form a team; the single player commands each Hero in turn.
* **Exploring:** Navigate through the dungeon to accomplish a Quest, which includes purging it of monsters, finding treasure, and solving puzzles. Search for secret doors, defuse traps, and avoid bands of patrolling monsters.
* **Negotiations**: Collect dungeon rumors from tavern patrons, negotiate and transact with non-player characters (NPCs) in town, and negotiate with hostile NPCs (monsters) to avoid battle in the dungeon.
* **Battle**: Fight against dungeon monsters: use weaponless fighting (overbearing, grappling, pummeling), missile weapons (e.g. bows and arrows, thrown daggers), and melee weapons (e.g. swords, flails, hand-held daggers). The avatar can even throw furniture and other items at enemies!

The Hero of *Adventurer* 1.0 can enter the various places to pursue the following activities. The examples are from the “out-of-the-box” *Adventurer* game.

* **Town Square**: Create and register a new Hero, and collect up to four Characters for team play. Newly generated Characters start in the Town Square of the village of Biljur’Baz, and are registered by the Bartleby the Scribner.
* **Tavern**: Collect information about the dungeon from NPC patrons, either by friendliness or bribery. Food and drink are also obtainable. The local tavern is named “The Ugly Ogre Inn,” run by Bork the Innkeeper. Patrons randomly move in and out of the tavern during the evening with different information, and willingness to divulge that information.
* **General Store**: Sell items the Hero no longer wants or needs, and buy items that the Hero wants. Sometimes the Hero can make a profit by selling items found in the dungeon. Biljur’Baz’s general store is named "The Rat's Pack", and run by the opportunistic merchant Dewey Howe.
* **Guilds**: Join a particular professional Guild and get an enhanced Character. A player's Character starts in earnest after reaching 200 XP and becoming a member of one of the Guilds: The *Arena* for Fighters, the *Monastery* for Clerics, the *Arcaneum* for Magic-Users, and the *Thieves’ Den* for Rogues. The Hero can be promoted to higher levels of ability in their Guild, and gets Quests from their Guild Master.
* **Dungeon**: Navigate the labyrinth of catacombs, ruins, castles, and other medieval venues to achieve a quest. Explore, battle monsters, find treasure, solve puzzles, and gain experience to rise to the next level of ability. The initial *Adventurer* has two starting dungeons just outside of the village of Biljur’Baz. (1) *Quasqueton*, a fairly large castle, contains a wide variety sampler of puzzles, magic, monsters, traps, and treasures, and enables the Hero to experience many of the features of the game. (2) The other dungeon, "The Temple of Pomarj" is intended for a more experienced player and has many intricate traps, and a few puzzles. This dungeon embodies the “segmented key adventure” style, in which various pieces of device must be assembled to achieve the final goal.

For example, the Character Falsoon is a member of the Fighter's Guild, and gets the Quest from Aragon the Guild Master to "find and retrieve Salazar's magic purple spear," known to be secreted away in Quasqueton. Falsoon buys supplies and rations from the "The Rat's Pack" using the few gold coins he has left. He grabs some dinner at "The Ugly Ogre Inn" and talks with a few of the patrons there, obtaining (some questionable) information. He then enters Quasqueton at dusk to achieve his quest. During his adventure, he garners clues, solves puzzles, fights monsters, avoids patrols of evil marauders, and obtains secret treasure, all getting him closer to his goal. Eventually, Falsoon finds the spear and returns it to the guild, richer in experience and gold. He gains more power and abilities as a result of his experience.

## Creating a New Character

A Character is generated once for the game and used in subsequent games, increasing in ability and power as a direct result of the quality and quantity of play. Generating a Character involves a collection of dependencies, so the Character must be generated in the order described below.

Each Person contains three internal objects: Race, Klass, and Inventory. Person will create these internal objects, and they will be responsible for any attributes specific to those objects. Attributes that are not specific will be generated by the Person object.

Generating the new Character, or Hero, follows a sequence of three steps: Collect the player’s input attributes, generate all the attributes that are used internally, and output the Hero’s profile. The profile is a reflection of the Hero’s internal attributes but also contains information that is inferred at the time the Profile is displayed; that is, they are modified or generated from internal attributes by the CIV component, but not stored as an intrinsic part of the Hero object.

See the respective sections *Collect the Player’s Input, Generate the Hero*, and *Display the Hero’s Profile*.

## General Design Layout

Both *Dungeon Wizard* and Adventurer have a similar design. The computer screen is split in two: the left side for text dialog, the right side for graphics. The basic unit of the dungeon is the *Room*, a playing area that is shown on the right side of the screen. Rooms can represent one or more indoor physical rooms, or outdoor areas of roughly the same scale. The floor plan is a 2-D view from above, and includes architectural structures like doors, stairs, and exits. The author builds floor plans using drag-and-drop actions from palettes of doors, furniture, monsters, and other items commonly found in the dungeon. Authors can invent new items, icons, skills, or monsters to add to the palettes for further customization.

## Architecture and Software Architecture Driving Principles

The application is driven by three critical design principles.

* **Model-Driven Architecture (MDA) and Separation of Concerns**. The application is broken into five primary components:
  + *Problem Domain Component* (PDC) that contains the core program (business) logic;
  + *Data Management Component* (DMC) for data persistence and management implementation;
  + *Human Interface Component* (HIC) for user interaction and GUI display;
  + Component Interface Validator (CIV) is a gateway between the GUI and the PDC, inserted to minimize the amount of logic in the GUI, which maximized automated testing.
  + *System Interface Component* (SIC) for communicating to external systems, across a network, through files, or other means.

Each component is further decomposed into responsibility-based objects. Each component talks with each other through a minimal communication "pipe", or data transport protocol. In the case of this application, that protocol is an EnumMap data shuttle. Each component's knowledge of each other is minimized to decrease coupling, and enable quicker maintenance. (The DMC and HIC are prohibited from talking to each other directly. The HIC can only talk through the CIV. Importing an HIC class in the PDC or DMC is forbidden, and vice versa. For more details, see each of the descriptions for the respective component packages.

* **Test-Driven Design (TDD) and JUnit automated testing**. JUnit does not test GUI code well, so a socket is needed to hook JUnit into, minimizing the amount of GUI code. The CIV allows this to happen, hiding only about 15% of code that cannot be tested with JUnit automation. The CIV is placed between HIC objects, particularly GUI widgets, and PDC objects to provide semantic validation of the GUI, and to allow a place to start and end JUnit integration tests. JUnit test flows start at an input CIV just behind an HIC's input GUI widget, and verify results collected from an output CIV just behind an HIC's output GUI widget. For more details, see the descriptions for the CIV package.
* **Data Shuttle Transport**. In order to pass PDC-independent data to and from the HIC, which know nothing about PDC type information, the GUI widget field data (always Strings) are placed into a data shuttle to the CIV. The data shuttle, after any necessary manipulation needed to prepare it for the PDC object, is passed to the PDC object. Each PDC object that communicated to a GUI widget has an unpackShuttle() method to move Strings from the shuttle and place the data into internal PDC format. Although the CIV can do this directly, it would require the PDC object to contain setters and getters for each attribute, which is not needed with the data shuttle concept. Getter and setter methods, which expose private variables to public scope and violate critical concepts of object-oriented data encapsulation, are avoided. Symmetrically, each PDC object that needs to output data has a method to create an output CIV and a packShuttle() method to place internal attributes into Strings in the shuttle for an output GUI widget. The data shuttle is implemented as an EnumMap<enum, String> where the key-value pair of the map is the enum key-field datum pair for the map.

## Attribute Types and Issues

### Attribute Types

Attributes, in general, are data characteristics of an object.[[1]](#footnote-1) There are several kinds of attributes at the foundation of both *Dungeon Wizard* and *Adventurer*. The attribute types determine how the data within an object is used, displayed, and interact with the various architectural components during execution of the application.

* *Direct attributes*: Atomic data that is specifically created and stored for a PDC object. Direct attributes are stored in internal format (i.e., most efficient form) as data elements of the object and may be referenced for display or used as the basis for calculations and inferred attributes. Direct attributes are not changed frequently, if at all. Ex: weight and height of a PDC Race object.
* *Inferred attributes*: Data that is created as the result of multiple direct attributes. Inferred attributes are usually not saved, but may be saved for expedience. If the base direct attributes change, then the inferred attribute must be re-inferred (recalculated) before use. Ex1: The Hero’s physical description is inferred from the direct attributes of weight, height, race, hair color, and charisma. Ex2: Action Points change infrequently, but are used frequently, so have a saved data element for convenience.
* *Translated attributes*: Data that is re-formatted for display purposes from internal format to display format, or vice versa, by the CIV. Ex1: Weight, internal format ounces, is translated to pounds and ounces for display. Ex2: Age, internal format seconds, is translated to years (integer format) for display.
* *Dynamic attributes*: Used in the widest sense of the term, dynamic attributes are calculations that characterize something about an object, and are never stored. They change so frequently, or depend on other high-frequency attributes, that a re-calculation is usually required each time they are needed. Simply stated, dynamic attributes are inferred attributes that are intentionally not stored. Ex: A Person’s speed results from their Action Points (inferred), weightCarried (inferred), weightAllowed (inferred), Race (direct), and the Race’s baseMovement (direct).

### Issue

It has always been an issue at the design level whether to save a calculation in a data element (inferred attribute) to avoid “recalculation next time” or recalculate the value each time is it needed. The answer depends on expediency, and how frequently the direct attributes are changed. (At the analysis level, the rule is that no inferred data should ever be saved if it represents a calculation, but that is for validation proofing reasons.) Inferred attributes that are saved can get “stale”, that is, the derived value is no longer accurate because its direct attribute(s) changed.

### Recommendation

Treat an inferred attribute as a direct attribute at the object’s public interface. The decision to save or calculate within the object is easier to determine, and encapsulates any chain of calculations within the object. A client object will not know, or need to know, if an attribute is inferred or direct, and can treat all public interface values as direct. With that structure in mind, no inferred attributes will appear to exist from a client’s perspective.

### Example: Person’s Speed

A Person’s speed is a dynamic attribute that is calculated and packed by the Person PDC object before sending the value to the CIV. The Person’s speed depends on

* Action Points, which depends on STR and DEX; and
* Base Movement, which depends on Action points and Race (unchanging); and
* Weight allowance, which depends on STR and weight of the Person (unchanging); and
* Load, which depends on total weight of Inventory, which depends on number of Items the Person is carrying.

If the STR, DEX, or number of items carried changes, then the speed of the Person ultimately changes. For convenience, if actionPoints is calculated as needed for Person, and load is recalculated and saved each time an Item is added or dropped, then calcSpeed() method, when called from a client, invokes the methods are shown in the diagram below.

Person.calcSpeed() -> Inventory.load();

-> Person.wtAllowance() -> Race.baseMvmt() -> Person.AP()

where the methods actually calculate, or merely access, the inferred attribute, depending on whether the owning object pre-calculated it or not; the client neither knows nor cares. It is also a fairly simple matter to change an object’s inferred attribute from a saved calculation to a recalculated one (or vice versa) if desired later.

## Future Versions

Future versions will support multiple players controlling their Characters on their own computers across the Internet. Characters and Teams can be created collaboratively, and team features will be enables, such as chat messaging and exchanging weapons between Players.

## Related Documentation

See the analysis documentation, use case requirements, and class diagram and sequence diagrams for the major use cases. See also the documentation for *Dungeon Wizard*.

# Collect the Player’s Input

*Adventurer* displays a set of options for collecting player’s input to generate a new Character. Five attributes are selected by the Player to create his or her Character, the rest are generated but only some are shown to the Player. The visible attributes are called the Hero’s Profile. The section below explains briefly each of these Character attributes.

The user is prompted to submit five pieces of information: Hero’s name, gender, Race, occupation, and hair color, as described below. The Options available to the player are also described. After the new Character, the Hero, is created, all input data is displayed in the Hero’s profile, in addition to the other attributes generated by *Adventurer*. See the section *Displaying the Hero’s Profile*.

**Name:** The Player is asked to enter free-text to define a name for the Hero. Aragon or Balthazar are much better names than Fred or Betty. The Hero’s name is the default filename into which the Hero data is stored, although the user may change it. Spaces are allowed in the name.

**Gender:**  The Character is male or female (radio-buttons). Female characters tend to be slightly weaker, shorter, lighter, more charismatic, and have a higher constitution than the males of their race.

**Hair Color:** This is a cosmetic option the player may choose, including bald. From this, and the Hero’s Charisma, height, and weight, the Hero’s appearance is inferred. The various options are shown in the list below.

Hair color options: {"black", "brown", "red", "blonde", "white", "gray", "streaked", "silver", "bald"};

**Race:**  The Character may select one of many races: Human, Dwarf, Elf, Gnome, Half-Elf, Hobbit, or Half-Orc. All traits and modifiers are based on the more common Human Character, but compensating restrictions and bonuses are applied to non-human Characters.

The table below shows the Races and Classes currently used in the game. In Chronos, there are no invalid Class/Race combinations. However, the user cannot select a Class when the Hero is generated: all new Heroes are created as Peasant Class. When the Hero joins a Guild, he or she becomes the Class of that Guild, never to go back to Peasant again.

Table Race and Class Options

|  |  |
| --- | --- |
| **RACE** | **CLASS** |
| Dwarf | Peasant |
| Elf | Cleric |
| Gnome | Fighter |
| Half-Elf | Magic User |
| Half-Orc | Thief |
| Hobbit |  |
| Human |  |

**Occupation:** The Hero most likely had a background of honest work (or at least a vocation) before becoming an Adventurer. The Player chooses one of the occupation options, which brings with it Skills that can be used during play. No Occupation is also an option, which will have no skills associated with it. Certain occupations have inventory that might be useful in other ways.

Each occupation is associated with a Skill; each Skill has a description and ability to add to the Hero’s bag of tricks. The player may choose from one of the options in the table below. For example, a Gambler has a higher-than-usual share of Luck; an Armorer is useful for field repair.

The Occupations must be selected from the Occupation Registry before being able to prompt the user to select one.

**The Skills are not shown in the options. They are inferred from the Occupation and are displayed after the Hero is generated.**

For more detail, see the section of *Adding the Hero’s Skills*.

Table Occupation Options (with Associated Skills)

|  |  |  |  |
| --- | --- | --- | --- |
| **Occupation** | **Skill** | **Occupation** | **Skill** |
| Acrobat | Tumbling | Innkeeper | Sense Motive |
| Alchemist | Arcane Knowledge | Jeweler | Appraise |
| Apothecary | Natural Knowledge | Leatherworker | Leatherworking |
| Armorer | Armor Repair | Mason | Find Secret Openings in Stonework |
| Banker | Brokering | Miner | Intuit Underground Direction |
| Bowyer | Bow Making | Sailor | Fast Swim |
| Carpenter | Find Secret Openings in Woodwork | Shipwright | Raft Making |
| Courtesan | Charm Person | Tailor | Sewing |
| Farmer | Predict Weather | Trader | Brokering |
| Fisher | Net Making | Trapper | Trapping |
| Forester | Intuit Direction | Weapon smith | Weapons Making |
| Freighter | Negotiations | Weaver | Appraise Tapestries |
| Gambler | Luck | Woodworker | Woodworking |
| Hunter | Hunting | None | No Skill |

When the player is satisfied, he or she submits the data. The player will next see the generated Hero’s Profile displayed containing the input data and attributes generated by *Adventurer* for the Hero.

# Generate the Hero

## Save the Player Input

The Character is a Person object, and the Player’s particular Player-Character is the Hero. *Adventurer* creates a Hero object by calling the Person object factory, passing the input data, which is then saved into the Hero object as direct attributes. In addition, the following steps must be followed to generate the remaining attributes of the Hero. The sequence of the steps is important because there are many dependencies between the Person, Race, Class, and Inventory objects that make up the Hero.

## Set the Initial Age

The Hero’s age is an value that depends on Race, created by the Age object. Each race has a different set of age brackets at which traits might change. Initial Heroes start at 85% of the youngest age bracket.

Table Starting Age by Race

|  |  |  |
| --- | --- | --- |
| **RACE** | **Age Brackets** | **Starting Age** |
| Dwarf | 50, 150, 250, 350, 450 | 42.5 |
| Elf | 150, 500, 800, 1100, 1350 | 127.5 |
| Gnome | 90, 300, 450, 600, 750 | 76.5 |
| Half-Elf | 40, 100, 175, 250, 325 | 34.0 |
| Half-Orc | 15, 30, 45, 60, 80 | 12.0 |
| Hobbit | 40, 125, 210, 325, 450 | 34.0 |
| Human | 20, 40, 60, 90 120 | 17.0 |

## Set the Prime Traits for Peasant

The *Prime Traits* define the Character’s basic makeup: Strength, Intelligence, Wisdom, Constitution, Dexterity, and Charisma, or as they are frequently abbreviated: STR, INT, WIS, CON, DEX, and CHR. Each Race contributes possible adjustments to these traits. The Prime Trait value is the base comparison against random values used to decide actions and effects. Each individual trait and use is explained below. Each trait also has secondary derived attributes that allow a more fine-grained set of abilities in play. See the section *Set Trait Modifiers*.

Each Character Class[[2]](#footnote-2) has a Prime Trait that is the key trait for that Klass, called a *Prime Requisite*. If the Character of a particular class does not have at least a 14 value in its Prime Requisite, the Character will never be very good in that Class. The Player should think about generating a new Character or changing the Class of the Character. See the section *Applying to a Guild* for more on prime requisites.

**Prime Traits are not shown in the profile**. See *Displaying the Hero’s Profile.*

**Strength (STR):** The physical power of the Character, and the Prime Requisite for the Fighter. Strength enables a higher chance of doing damage in battle, carrying heavier loads, kicking in locked doors, and bending bars or lifting heavy gates. It is one of the major components in hand-to-hand fighting. All Characters have STR modifiers.

**Intelligence (INT):** The mental power of the Character, and the Prime Requisite for the Magic-User. Intelligence is used by all Characters to identify items, and reflects the amount of knowledge of the Character. Only Magic Users have INT modifiers.

**Wisdom (WIS)**: The spiritual power of the Character, and the Prime Requisite for the Cleric. Higher WIS allows a higher chance to resist magical attacks. Clerics have additional WIS modifiers.

**Constitution (CON):** The physical robustness of the Character. It is important to all Characters, but not a Prime Requisite of any one Class. This trait reflects the Character’s ability to withstand physical trauma or poisons. It also adds to the Hit Points (life quantifier) of the Character at Guild promotions. All Characters have CON modifiers.

**Dexterity (DEX):** The natural quickness or agility of the Character, and the Prime Requisite for the Rogue (or Thief). DEX is also used to modify defensiveness in combat, and to hit with a missile weapon. DEX is a major component in non-lethal fighting. All Characters have DEX modifiers.

**Charisma (CHR):** The charm or leadership qualities of the Character, and ability to persuade others. CHR is used heavily to benefit the Character or his party during Negotiations, such as avoiding battle or buying and selling items at the General Store. There are no CHR modifiers.

All Peasants start with a fixed mediocre set of Prime Traits until they join a Guild. Although traits are Klass attributes, the six Prime Traits are set to 11. Gender, age, and Race traits apply afterwards. When the Peasant joins a Klass Guild, then the traits are re-evaluated using the “4d6-1 Rule.” See *Applying to a Guild*.

## Adjust Traits for Gender

Gender is a Race-based attribute. Male characters are the default; female Characters get -1 STR, +1 CON, +1 CHR. Race limits override Gender adjustments.

## Adjust Traits for Age

Age is a Race-based, but it comes from the Age object. As Heroes cross the age bracket, their traits are adjusted. All new Characters are (by definition) “Young Adult”, and therefore get -1 WIS and +1 CON. Race limits override Age adjustments.

## Adjust Traits for Race.

Some non-human races get adjustments to their rolled traits. The table below summarizes how traits are adjusted for each race.

Table Trait Modifications by Race

|  |  |
| --- | --- |
| Dwarf | CON+1, CHR-1 |
| Elf | CON+1, DEX+1 |
| Gnome | No modifications |
| Half-Elf | No modifications |
| Half-Orc | STR+1, CON+1, CHR-2 |
| Hobbit | STR-1, DEX+1 |
| Human | No modifications |

## Verify the Race Limits

If the Trait being verified (after age, gender and race adjustments) is less than the minimum, or greater than the maximum, then the trait value is set to the minimum or maximum respectively. The table below shows into which trait ranges each Race trait must fall. Traits may drop or rise below or above the limits during play, but these values are defined for generating a new Peasant Character, and are used for promotions. Race limits override age, gender, and race adjustments.

Note that no trait is above 19 or below 6 for any of the Races.

Table Range Limits for Races

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TRAIT** | **Dwarf** | **Elf** | **Gnome** | **Halfling** | **Half-Elf** | **Half-Orc** | **Human** |
| STR | 7-17 | 6-16 | 6-15 | 6-14 | 7-18 | 10-19 | 8-19 |
| INT | 8-18 | 8-18 | 7-18 | 8-18 | 8-18 | 7-17 | 8-18 |
| WIS | 8-18 | 8-19 | 8-18 | 7-17 | 8-18 | 6-14 | 8-18 |
| DEX | 7-17 | 9-19 | 8-18 | 8-18 | 9-18 | 7-17 | 8-18 |
| CON | 12-19 | 8-18 | 10-19 | 10-19 | 8-18 | 13-19 | 8-18 |
| CHR | 6-16 | 9-18 | 7-18 | 8-18 | 8-18 | 6-12 | 8-18 |

## Find the Height and Weight

**Weight (in ounces**): The naked weight of the Character without any inventory. Heavier Characters get overbearing and grappling bonuses when fighting. Ounces are used internally for more accurate comparisons, but displays are shown in lbs.

**Height (in inches):** The height of the Character. Height is used to determine what the Character can reach, and where something might hit the Character (or go over his head). Short Characters (with a height =< 46 inches) get a –1 Speed penalty and tall Characters (with a height >= 78 inches) get a +1 Speed bonus.

Weight and height attributes are calculated randomly from a racial average and adjusted for gender. The Hero’s weight and height are calculated from a normal (Gaussian) distribution such that each attribute will fall within one standard deviation of the mean. The calculations are organized to keep the population of Characters within a race at a characteristic height and weight ratio. Inches are used internally for more accurate comparisons, but displays are shown in feet and inches.

Table Character Weight Table (in pounds)

|  |  |  |
| --- | --- | --- |
| **Race** | **Male Average Weight** | **Female Average Weight** |
| Dwarf | 150 | 120 |
| Elf | 100 | 80 |
| Gnome | 80 | 75 |
| Half-Elf | 130 | 100 |
| Hobbit | 60 | 50 |
| Half-Orc | 150 | 120 |
| Human | 175 | 130 |

Table Character Height Table (in inches)

|  |  |  |
| --- | --- | --- |
| **Race** | **Male Average Height** | **Female Average Height** |
| Dwarf | 48 | 46 |
| Elf | 60 | 54 |
| Gnome | 42 | 39 |
| Half-Elf | 66 | 62 |
| Hobbit | 36 | 33 |
| Half-Orc | 66 | 62 |
| Human | 70 | 64 |

.

## Calculate the Modifiers for Each Prime Trait

The Prime Traits have *Modifiers*, additional properties implied as a result of a certain Prime Trait value. These *modifications* are not the same as *adjustments* (such as for gender or race) that change the Prime Traits themselves, but are additional, derivative attributes for the Hero.

After making Race, Age and Gender adjustments, and calculating the height and weight of the Hero, set the Trait modifications according the following tables. Traits less than 6 (the minimum for any of the Races) and greater than 19 (the maximum for any of the Races) are included for cases where the Character changes his or her Prime Trait due to some circumstance during play.

### Assign Strength Modifiers

The Strength modifiers are inferred attributes, and must be recalculated each time they are used. They are not used as part of the Character generation except to be initialized.

* **To Hit Mod:** The Character’s probability to “‘Hit” (achieve damage) is adjusted for non-missile weapons: extra strength allows a Fighter to get through armor more easily to do damage. The modifier sometimes depends on weapon used, but does not apply to missile weapons. The numbers shown are increases/deceases from a normal d20 roll (units of 5%) against opponent’s Armor Class.

*To Hit Mod* uses the following calculation:Hero gets an intrinsic +1 bonus for every point of STR > 16; that is, STR=17 gets +1 To Hit Mod. For every point STR < 8, mod gets a penalty of –1; that is, STR = 6 get –2 To Hit Mod. All other STR values have a 0 (normal) value for those this mod.

* **Damage Mod:** Extra or diminished damage due to the Character’s Strength when a Hit is scored; does not apply to missile weapons. The numbers shown are increases/deceases from a die roll for damage, which is weapon dependent.

*Damage Mod* uses the following calculation:Hero gets an intrinsic +1 bonus for every point of STR > 16; that is, STR=17 gets +1 Damage Mod. For every point STR < 8, mod gets a penalty of –1; that is, STR = 6 get –2 To Hit Mod and -2 Damage Mod. All other STR values have a 0 (normal) value for this mod.

* **Weight Allowance:** The amount of weight, in pounds, the Character can carry without being encumbered, that is, without losing Speed. See Encumbrance in the Playing section.

Weight allowance is calculated based on the weight of the person. Each Hero can carry their own weight, but not more, adjusted by their STR, without being encumbered. Weight allowed is the ratio of STR / average STR (11.5) \* weight of the person.

Example 1: A 150 lb. person with a STR=16 can carry [16/11.5 = 1.39] \* 150 lb. = 208.7 lb. without encumbrance.

Example 2: A 200 lb. person with a STR=11 can carry [11/11.5] \* 200 lb. = 191.3 lb. without encumbrance because they are weaker than average.

### Set Languages from Intelligence

Characters can learn languages based on their intelligence and time. Rogues have a special Guild language, *Thieves’ Cant*, which they must learn before being promoted to Level 2. See *Languages* and *Promotions*.

* **Languages**:Peasant Characters know the *Common* language, the *lingua franca* of the area that everyone can speak, and the language of their race (if not Human). Half-Orc and Half-Elf Heroes have a 50% chance of knowing their race language.

Table Race Languages

|  |  |
| --- | --- |
| Dwarf | Dwarven |
| Elf | Elvish |
| Half-Elf | 50% chance to know Elvish |
| Gnome | Gnomen |
| Hobbit | Tolkeen |
| Half-Orc | 50% chance to know Orcish |

* **Maximum Learnable Languages**: The total number of additional languages the Character can learn in his or her lifetime, given an appropriate teacher and enough time, depending on INT. The number of learnable languages does not include the Common language or the racial language for non-human Characters. See Table below.

Except for language ability, other Character’s Intelligence modifiers apply only to Magic Users, which are described in the section *Applying to a Guild.*

Table Maximum Learnable Languages by Intelligence

|  |  |  |  |
| --- | --- | --- | --- |
| **INT** | **Max Langs** | **INT** | **Max Langs** |
| 8-9 | 1 | 14-15 | 4 |
| 10-11 | 2 | 16-17 | 5 |
| 12-13 | 3 | 18-19 | 6 |

*Algorithm for this table:* Learnable languages = INT/2 – 3; (integer arithmetic).

NOTE: Learning a language means that the Hero can speak and understand it. He or she must have the proper Intelligence to be *Literate* and be able to read and/or write it.

### Assign Wisdom Modifiers

* **Magic Attack Mod**: Wise people are more resistant to magic attacks, and get an intrinsic +1 bonus for every point of WIS > 14; that is, WIS = 16 gets +2 Magic Attack Mod. For every point WIS < 8, Magic Attack Mod gets a penalty of –1; that is, WIS = 6 get –2 Magic Attack Mod. All other WIS values have a 0 (normal) Magic Attack Mod.

Dwarves, Gnomes, and Hobbits get another Magic Attack Mod bonus to what they get from their WIS because of their natural magic resistance. See the Constitution Modifiers.

### Assign Constitution Modifiers

* **Hit Point Modifier (HP Mod)**: People with strong constitutions are harder to kill (which is measured in Hit Points). Higher CON values get a bonus of +1 for every point that their CON > 14; that is, CON = 16 gets +2 HP Mod. For every point CON < 8, HP Mod gets a penalty of –1; that is, CON = 6 get –2 HP Mod. It is also sometimes used to absorb poison effects. HP Mod is also used to calculate additional HP at promotion time, so it is a Klass attribute.
* **System Shock Percent (SS)**: The percent chance that the Hero does not react adversely to adverse physical situations. For example, a Hero with a 48% SS that jumps into ice-cold water has a 48% chance of not getting cramps and sinking like a stone. (SS for CON 13 and 14 are only 2 percentage points apart).

*Algorithm*:

* For CON < 14, the Character’s SS = (CON+4)\*5; that is, CON=10 implies SS = 70%.
* For 14 <= CON <= 18, SS = 87+ 3\*(CON-14); that is, CON=14, SS=87%, and CON=16, SS = 93%.
* For CON > 18, SS = 99%; no Hero can have 100% resistance to bodily trauma.

***Should this be in the spec, or should system shock be calculated from CON directly, as in Save vs CON-2 situations? For now, this is removed from the spec until known to be needed (YAGNI).***

*Dwarves, Gnomes, and Hobbits Only*

These Races get an additional racial adjustment to Magic Attack Mod because of their natural resistance to magic, reflected by their CON.

* **Additional Magic Attack Mod**: These races get an additional +1 per 4 points of CON because they are resistant to magic. For Dwarf with WIS=15, CON=15, set Magic Attack Mod to +4 (+1 for WIS and +3 for CON).

### Assign Dexterity Modifiers

New Characters have no armor, shields, or weapons. When they buy equipment, they get an Armor Class (defensive measure) equal to the kind of defensive equipment they buy. The To Hit Mod and Armor Class mod are natural adjustments due to speed and agility.

* **To Hit Missile Mod**: For each DEX > 14, add +1 to hit with a missile weapon; for each DEX < 8, add –1 to hit with a missile weapon. For example, DEX=15 implies To Hit Mod = +1.

*Elves Only*

Elves get an extra +1 To Hit with missile weapons, which is applied only when the Elf is wielding a missile weapon.

* **Armor Class Mod (AC Mod)**: For each DEX > 14, add +1 to AC (defense from being hit) because of superior ability to avoid the attack; for each DEX < 8, add –1 to AC Mod. For example, DEX=7 implies AC Mod = -1. There is no AC mod for new Characters.

### Assign Charisma Modifiers

Charisma is used for buying and selling, and for success during negotiations, such as getting information from Tavern patrons. Depending on Charisma, the Hero will get a lower or higher price when buying and selling, or the Hero’s Save is adjusted by that amount when negotiating.

There are no Charismamodifiers.

## Initialize Derived Attributes

### Calculate Initial Hunger State

The Character must eat regularly or die of starvation. Each Character requires 15 calories per pound of weight per day, measured in satiety points, or SP, and burns SP during play. When the Hero’s satiety points go to zero, calculated to happen after three days, the Hero will begin to get weak, and lose STR points until dead (or eats something). Although each edible item must have associated SP, it also opens the dungeon to provide poisons, and an incentive for a hungry Hero to risk eating or drinking them.

When the Hero is created, he or she is FULL. The table below shows the percent of SP the Hero has before changing from one Hunger state to another, although this is not calculated until play, when the Hero is performing actions.

The Hunger object, given the Hero’s weight, will calculate the required satiety points per day for the new Character, and the rate at which calories are burned.

Different actions performed burn satiety points at different rates. For example, combat will burn satiety points at triple the normal rate, sleeping and resting burn them at half rate.

The secondary effects of a weakened Hero will mean all STR mods will decrease, particularly To Hit and Damage Mod penalties, encumbrance when moving under load, and eventually, when the STR drops to 3, the Hero will become unconscious. At STR = 0, or when the Hero is STARVED, the Hero is dead.

Table Hunger States for Satiety Point Ranges

|  |  |
| --- | --- |
| **Hunger State** | **SP Range (%)** |
| FULL | [90, 100] |
| NOT HUNGRY | [30, 90) |
| HUNGRY | [0, 30) |
| WEAK | [-40, 0) |
| FAINT | [-100, -40) |
| STARVED | < -100 |

Note: Square brackets indicate inclusive boundaries; parentheses indicate exclusive boundaries.

### Set Level and Experience Points

Level and Experience Points (XP) indicate promotable benchmarks for the Hero, and are therefore Klass attributes.

**Level:**  The large-scale unit of measure of the Character’s prowess and abilities. All Characters start at Level 0. After 200 XP, they can join a Klass Guild, and are promoted to Level 1. (Although NPC’s may be Assassins, Player Characters may not.) Character levels range from 0 to 10. See *Promotions* and *Table 21. Experience Points Required by Level.*

*Caution:* Dungeons and monsters are rated with Levels too, but these are not strictly related to Character Level. Dungeon and monster levels indicate a relative hierarchy of difficulty in overcoming them. For example, a Level 4 dungeon will have monsters ranging from Level 2 to Level 6; a Level 1 dungeon will have monsters ranging from Level 1 to Level 3.

**Experience Points (XP):** The small-scale unit of measure for the Character’s prowess or life experience: the more experience in the dungeon, the more experience points. XP are awarded for killing monsters, finding treasure, solving puzzles, and risky (but not foolhardy) play. As the Character gains XP, they are eligible for promotion at their Guilds, at which time their abilities and power rise too. See *Promotions* and *Table 21. Experience Points Required by Level.*

All new Heroes start at Level 0 and XP = 0.

### Set Hit Points

**Hit Points (HP):**  Hit Points are the degree of life left in the Character, who gains and loses HP while playing. Damage from attacks and other unfavorable events decrease the Character’s HPs. Starting at Level 0, each Character is given an amount associated with his class (Peasants start at 10 HP), then are given more at each Guild promotion. See *Table 11. Starting Hit Points by Class*. See *Promotions* in the *Playing* section for obtaining HP, and other benefits, for higher levels*.*

* If the Character’s HP decreases to 0 but not less than 0, the Character is unconscious.
* If the Character’s HP becomes lower than 0, the Character is dead. Period. No resurrection. Out of the game.
* Characters regain 1 HP per 12 hours of rest (sleep or inactivity). Doctors or magic may also be used to increase HP more quickly.

All new Peasants start with 10 Hit Points, adjusted by their HP Mod.

### Set Initial Gold

“Gold” refers to the cash of the milieu, and consists of gold pieces, silver pieces, and gems. Ten silver pieces (sp) = 1 gold piece (gp). All new Heroes start with a little cash in their pockets, but starting cash for Peasants is less than any Guild member would get. The poverty level of the new Character will require him to constantly fund himself through dungeon exploration for treasures, looting, and killing monsters.

If the Hero becomes wealthy, or wants to stash some away in case he gets robbed, the Hero can put cash, valuables, and other Items in the bank, which is always retrievable (although there is a small bank fee). The Hero can also set another Hero as a beneficiary in case of his death. Usually this device is used to allow wealth and items to transfer from one on the player’s Characters to another in case the player loses one of his Heroes.

Some gold and silver pieces are given as part of the default inventory given to a new Character. The Hero can purchase a few supplies from the General Store in town with the slight cash he has.

Set the new Hero with gold and silver according to the default Inventory shown in Table 11. Set the Hero’s Gold Banked = 0 gp / 0 sp.

## Set Starting Inventory and Related Attributes

All new Characters get some beginning inventory because they are coming from someplace and doing something. The inventory given is the bare basics, and the Hero can augment that slightly with the little cash they start with.

Inventory is calculated before the traits like Action Points and Speed because they depend on the items or weight of the inventory.

Each new Hero gets the following before joining a Guild, at which time they get fully equipped.

Table Beginning Inventory and Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Item** | **Weight (oz)**  **per Item** | **Quantity** |
| General | Backpack | 160 | 1 |
| General | Cloak | 32 | 1 |
| General | Belt | 5 | 1 |
| General | Belt Pouch, small | 2 | 1 |
| General | Breeches | 16 | 1 |
| General | Pair of Boots | 40 | 1 |
| General | Shirt | 8 | 1 |
| General | Tinderbox, Flint & Steel | 5 | 1 |
| General | Torches | 8 | 3 |
| Provisions | Rations | 2 | 3 |
| Provisions | Water skein | 80 | 1 |
| Weapon | Quarterstaff | 48 | 1 |
| Cash | Gold | 2 | 15 |
| Cash | Silver | 1 | 8 |

The total weight of all items that Hero carries is 464 ounces (29 lbs).

The Inventory is a Registry object that contains all Item objects organized by category. The Cash does not show in the standard Inventory list, but has a special place in the attributes section because it is used so frequently.

**The AP modifiers below (pummeling, overbearing, grappling, and shield bash) are not created when the Hero is generated, but each time they are used during runtime.**

### **Calculate Action Points and Modifiers**

**Action Points (AP):** The success of many actions a Character takes is determined by the Character’s Strength and Dexterity, so AP = STR + DEX as a convenience. AP is a property of the Character and never changes unless the STR or DEX of the Character Change.

The AP determines the Speed of the Character, results of hand-to-hand combat (pummeling, overbearing, grappling, and shield bash), and other activities decided by combinations of Strength and Dexterity.

The AP modifiers depend on inventory and weight carried, in addition to the Hero’s physical weight, so it must be recalculated each time just before it is being used. When creating a Hero, AP modifiers depend on Inventory items.

*Non-lethal fighting* (e.g. hand-to-hand combat) is so common that standard modifiers are given as a convenience during melee. The modifiers below apply to d20+AP rolls instead of d20 against AC. See *Non-Lethal Fighting* in the *Playing* section.

* **Overbearing**: The charge of one opponent to another with intent to tackle him or knock him back 5 ft. Overbearing can be preceded by a Shield Bash. The modifier reflects weight advantages between two opponents.

Set this modifier equal to AP, and +1 for each 25 lb. of weight of the Character. If Character buys metal armor, then this value increases by +4.

* **Grappling**: The manhandling or wrestling of two opponents to hold or coerce the opponent into a worse situation. The modifier reflects possible armor, e.g., metal gauntlets are not dexterous and pose a disadvantage.

Set this modifier equal to AP + Damage STR modifier + To-Hit DEX modifier. If Character buys metal armor, then this value decreases by -4.

* **Pummeling**: The striking or bludgeoning of an attacker to knock the other unconscious or subdue. The modifier reflects possible armor, e.g., pummeling with a metal gauntlet gives an advantage.

Set this modifier equal to AP. If Hero buys metal armor, then this value increases by +4.

* **Shield Bash**: A pummeling technique to knockdown or knock back an opponent, to disrupt his turn’s action, or to prevent an attack of opportunity. The Hero must have a shield for this value to apply.

Set this modifier equal to 0 if the Hero has no Shield, else to AP if the Hero gets one.

### Set Armor Class (AC)

Each Character starts with an AC of 10 (no armor, 50% chance of not being hit) that increases with any armor worn. AC defines the minimum value required before the Character takes damage from a hit. AC can also be increased with magic items and Dexterity modifications (AC Dex Mod). AC = (10 + armor bonus) + shield bonus + Dexterity modifier + size modifier. For example, plate mail + shield + 1 (Dex to Hit) + 0 (size) = 16+1+1+0 = AC 18. A list of possible armor classes by armor type is given in the *Table. Armor for Fighters* in the section *Generating the Character*. However, there is *always* a possibility of damage from a Critical Hit even if AC > 20. See *Critical Hits and Fumbles.*

Armor class always starts at 10 + AC Mod (DEX modifier). The AC changes as armor is worn or removed. AC is an Inventory calculation because it depends strongly on the armor worn.

### Init Speed

**Speed is an inferred attribute calculated at the time the Hero is displayed, or during runtime. It is not a direct attribute of Person.**

How fast your Hero can move to attack or flee monsters, is shown as a number typically from 2 to 6. In special cases, the Hero may be slowed to a stop. Speed applies during melee, in which the distance a Character can move in 1 round (10 seconds) is important. The Hero’s speed is determined by a combination of Strength and Dexterity, amount of weight carried, and Character size (short dwarves usually move slower than tall humans). For racial adjustments, see *Section 2. Generating the Character.* For Characters that are carrying significant weight, burdened, or encumbered, their mobility decreases; see *Encumbering.*

As a starting point, (normal size unencumbered Character), set the Character’s base movement, where a unit is 5’ in any direction, from the table below. Typical base movements are 20ft/10 sec turn.

Table Block Movement by Action Points

|  |  |
| --- | --- |
| Action Points | Base Movement |
| 1 <= AP <= 15 | 2 |
| 16 <= AP <= 23 | 3 |
| 24 <= AP <= 32 | 4 |
| 33 <= AP <= 40 | 5 |

If the Character is less than 48 inches tall, subtract 1 BM from the table value. If the Character is more than 78 inches tall, add 1 BM to the table value. Block movement must be calculated during play because it depends on the Hero’s equipment and load.

## Add Skills and Special Abilities

The new Hero has four kinds of Skills: Literacy, Occupational Skill, Race-specific Skills, and Klass-specific Skills. The Peasant Klass has no skills, so only the first three are used during Hero creation. Although a Hero can own a Skill, any Skill assumes that the resources (tools and supplies) are available for that Skill to be used.

### Assign Literacy Skills

Most people in medieval days could not read or write. Adventurers tend to be better than average, so they might be able to do both. Characters who can neither read nor write means they cannot read or write Common either. If the Characters can read and/or write Common, then they have that skill for all their known languages.

Assign Literacy skill according based on the Hero’s INT:

* Hero can neither Read nor Write if INT < 10.
* Hero can Read but not Write if 10 <= INT < 12.
* Hero can both Read and Write if INT >= 12.

### Assign Skills from Occupations

Occupational skills allow the Character to make crude items in the field, always at a –1 or –2 quality. For example, an Armorer makes minor armor or weapons in the field but cannot work with *mithril* to make armor. If a Bowyer makes a bow, it will be –1 to hit; arrows will cause –1 damage. See special note on Kits below.

For the Hero’s occupation, select the associated Skill from the Skill Registry and add it to the Hero’s skill list. Some Skills require a minimum required trait. Add these Skills only if the Character satisfies the minimum Trait listed for that ability.

### Assign Racial Skills

The following sections list the Special Abilities (Skills) for the non-human races; humans do not have Special Abilities. See the section on *Playing* for how these Abilities operate.

NOTE: All non-human races have infravision ability for some distance. For convenience, Humans are therefore given an infravision skill of 0 ft.

###### Dwarf Abilities

* *Infravision 60’:* Ability to vaguely see warm bodies (infrared radiation) in the dark.
* *Geasing:* Intentional ability for underground or mining senses; must involve stonework within 10’
* Detect slopes in underground passages (75%);
* Detect new construction in tunnel (75%)
* Detect sliding or shifting walls or rooms (66%)
* Detect stonework traps (50%)
* Determine approximate underground depth (50%)

###### Elf Abilities

* *Resistance to Sleep Spells* (90%): If the d100 resistance roll fails, an Elf still gets a d20 < WIS + Magic Attack Mod to save.
* *Resistance to Charm Spells* (90%)*:* If the d100 resistance roll fails, an Elf still gets a d20 < WIS + Magic Attack Mod to save.
* *Infravision 60’:* Ability to vaguely see warm bodies (infrared radiation) in the dark.
* *Archery:* All elves get a +1 To Hit when using any kind of bow, except a crossbow.
* *Tingling:* ability to detect secret doors or concealed doors if within 10’, either actively searching (33%) or to notice in passing (16%).
* *Move Silently:* Elves at Level 0 have this ability equal to a first level thief of the same DEX. For example, Galdwel, DEX=15, can use this ability at 20%.

###### Gnome Abilities

* *Infravision 60’:* Ability to vaguely see warm bodies (infrared radiation) in the dark.
* *Geasing:* Intentional ability for underground or mining senses; must involve stonework within 10’
* Detect slopes in underground passages (80%);
* Detect unsafe walls, ceilings, floors (70%)
* Determine direction of underground travel (50%)
* Determine approximate underground depth (60%)

###### Half-Elf Abilities

* *Resistance to Sleep Spells* (30%): If the d100 resistance roll fails, a Half-Elf still gets a d20 < WIS + Magic Attack Mod to save.
* *Resistance to Charm Spells* (30%*):* If the d100 resistance roll fails, a Half-Elf still gets a d20 < WIS + Magic Attack Mod to save.
* *Infravision 60’:* Ability to vaguely see warm bodies (infrared radiation) in the dark.
* *Tingling:* ability to detect secret doors or concealed doors if within 10’, either actively searching (33%) or to notice in passing (16%).

###### Hobbit Abilities

* *Resistance to Poison*: If the d100 resistance roll against System Shock fails, a Hobbit still gets a saving throw of d20 < WIS + HP Mod to save. If the save fails, the CON deductions are taken from both CON + Magic Attack Mod value.
* *Infravision 30’:* Ability to vaguely see warm bodies (infrared radiation) in the dark.
* *Geasing:* Intentional ability for underground or mining senses; must involve stonework within 10’
* Detect slopes in underground passages (75%);
* Determine direction of underground travel (50%)

###### Half-Orc Abilities

* *Infravision 60’:* Ability to vaguely see warm bodies (infrared radiation) in the dark.

# Display the Hero’s Profile

The Hero’s Profile contains the player’s input data, some of the Hero’s intrinsic attributes, and user interface data (fields) *inferred* or *transformed* from the Hero’s attributes. Each is described below in three sections, and how to infer the specific fields from the attributes.

The HeroProfile object in the CIV component infers the output fields that are displayed in the Profile. All GUI data are of String type.

## Player’s Name Plate

Four of the five input fields are displayed directly, along with the default Peasant Klass. These five data elements are part of the “Name Plate” on the display screen.

* Name of the Hero
* Gender (male or female)
* Race (one of eight)
* Occupation (one of many).

The fifth input datum, hair color, is used to infer the Hero’s description, and is not shown in the Name Plate.

## Displaying the Hero Attributes

Beneath the Name Plate, the attributes needed for the Player to direct his or her Character are shown. Direct attributes are the same as internal Person attributes; otherwise they are translated from internal attributes to something else as a GUI String.

### Row 1: XP, Level, HP, Occupation, Hunger

Row 1 contains these attributes.

* XP (initialized to 0)
* Level (initialized to 0)
* Hit Points (initialized to 10)
* Occupation name (user selected)
* Hunger State (translated from SP, initialized to FULL)

Hunger state is translated from satiety points to one of the enum states:

*FULL*, *NOT\_HUNGRY*, *HUNGRY*, *WEAK*, *FAINT*, *STARVED*;

Based on the percentages given in the section *Generating the Hero,* the new Hero is always initialized to FULL.

### Row 2: AC, Speed, Age, Height, Weight

Row 2 contains these attributes:

* Armor Class (depends on inventory; initialized to 10)
* Speed (translated from double to dimensionless float 1.1)
* Age (translated from seconds (long) to years (integer))
* Height (translated from inches to feet and inches)
* Weight (translated from ounces to lbs and ounces)

### Row 3: Gold and Weight Carried

Row 3 contains these attributes:

* Gold and Silver pieces (direct)
* Gold Banked (direct; initialized to 0.0)
* Weight Carried (translated from ounces to lb and ounces)

### Row 4: Languages

Row 4 contains these attributes:

* Max Languages (direct)
* Languages known (direct list)

### Row 5: Hero’s Description

Row 5 contains the Hero’s inferred physical Description. Based on the Person's height, weight, hair color, and Charisma, generate a text description about that Person’s appearance. Above-average Charisma results in positive connotations of the Hero’s body type, and below-average Charisma gives a similar description but with negative connotation.

The description follows the template:

"A <height descriptor> and <weight descriptor> <gender> with <color> hair.

[She | He] is <CHR reflection>".

Light, Average, and Heavy weights refer to the Human standard for all races, so is based on the minimum, average, and maximum values for a Human. The description is given from the Human (Player) point of view; a tall gnome is still a short character to a Human.

Table Positive Height/Weight Descriptors

|  |  |  |  |
| --- | --- | --- | --- |
| **Weight/Height** | **Light** | **Average** | **Heavy** |
| **Short** | Petite | Compact | Burly |
| **Average** | Lithe | Athletic | Muscular |
| **Tall** | Thin | Tall | Towering |

Table Negative Body Type Descriptors

|  |  |  |  |
| --- | --- | --- | --- |
| **Weight/Height** | **Light** | **Average** | **Heavy** |
| **Short** | Tiny | Pudgy | Squat |
| **Average** | Slinky | Average-size | Heavy |
| **Tall** | Skinny | Tall | Giant |

Table Charisma Descriptors

|  |  |
| --- | --- |
| **Charisma** | **Descriptor** |
| < 8 | "crippled and horribly ugly" |
| 8 | "horribly scarred" |
| 9 | "scarred from war or fire” |
| 10 | "the result of years of misery |
| 11 | "weather-beaten and tough" |
| 12 | "nothing special to look at" |
| 13 | "clear-eyed and rugged but not handsome" |
| 14 | "slightly attractive if one could scrape off the years of wear and tear" |
| 15 | "a handsome adventurer” |
| 16 | "gorgeous" |
| 17 | "very attractive" |
| 18 | "stunningly beautiful" |
| > 18 | "mesmerizing, and you will do whatever this person commands of you” |

### Putting these fragments together creates a somewhat varied description of the person that automatically reflects that Hero’s Charisma and body type.

## Displaying the Hero’s Inventory

The Inventory is shown as a single row in a scrolling table, and grouped by Category, which is one of (in order) ARMOR, WEAPON, GENERAL, PROVISION, or VALUEABLE. CASH is also a Category, but since Gold and Silver pieces are shown in Row 3, it is not included in the Inventory list, but it is included in the Inventory weight.

For each Item, a single line displays:

* Category (direct)
* Quantity (direct)
* Item’s Name (direct)
* Item’s Weight (translated from ounces to lbs and ounces)

## Displaying the Hero’s Skills and Special Abilities

The special abilities and skills are shown as a list of Strings in a separate scrollable table below the Inventory. Almost every Character has some special ability because of their Literacy, Occupation, and Race. Later Klass abilities will be added when the Peasant Hero joins a Guild.

* Literacy: “Cannot read or write”, “Can read but cannot write”, “Can read and write”
* Occupation: List the Skill associated with the Occupation. Occupation = “None” means Skill = “No Skill”, but that should be inferred from the Occupation object itself.
* Special Abilities: The list inherited from the Race, but the subabilties are descriptive and are not shown. For example, Geasing comes into play several ways, but only “geasing” is shown; the sub-abilities are actually Skills that come into play later.

1. The word *attribute* is used in here in a slightly wider sense than a strict object attribute, which is a data element that helps define that object, and holds state information. [↑](#footnote-ref-1)
2. The Character *Class* is the Hero’s profession: Fighter, Magic-User, Cleric, or Rogue in version 1.0, but the word *Class* also refers to the implementation classes of the object model and implementation language. In cases where confusion may exist, the world *Klass* is used to mean the profession. [↑](#footnote-ref-2)