Pink Trenchcoat

a cyberpunk rule-set

Serbitar

June 3, 2023

Contents

Chapter 1

Basics

This chapter will cover the basics of Pink Trenchcoat including standard RPG nomenclature as wells as methods of conflict resolution. The rule system uses a fixed set of resolution methods, which are covered here, that will be used throughout the system exclusively.

1.1 Definitions

A couple of basic descriptions and definitions are given here.

1.1.1 Gamers

Everyone that is taking part in the game is a *Gamer*.

Game Master The *Game Master* is the person that is not playing their own *Character*, but all the *Characters* that are not being played by a *Player*.

Players A Player is a *Gamer* that is only playing their *Character* and maybe *Characters* that are closely connected to this *Character* like *Drones*, *Agents* or *Contacts*.

1.1.2 Characters

A *Character* is an entity that can actively make decisions in the game world and act on those decisions. In Pink Trenchcoat this includes (Meta)-Humans, but also *Agents*, *Drones*, *Spirits* and more.

Player Characters A *Player Characters* or *PC* is a *Character* that is directly and often exclusively controlled by a *Player*.

Non-Player Characters All *Non-Player Characters* or *NPC* are most often controlled by the *Game Master*.

1.1.3 Mathematics

Pink Trenchcoat's resolution system only uses integers. Although during calculation a number mit be not an integer, it needs to be rounded to the next integer for any kind of *Test*.

Rounding Fractions are always rounded mathematically correct. This means that 0.5 is rounded to 1.

1.2 Dice

Like most game systems Pink Trenchcoat uses dice to act as a randomizer for *Tests*. This is done to increase tension during the game session and include a random element so that players can not plan everything in advance with 100% certainty. However, if the gaming group so chooses, the rule set can be used completely without dice, as the average result of a die roll is always 0.

Pink Trenchcoat uses five six-sided dice with two "-", two blank and two "+" symbols also known as FUDGE dice. They are always used together and there are no other dice rolls used.

Almost always a player will roll only 5 dice, and the game master will secretly roll the other 5 dice, either because its an *opposed test*, and the game master is performing the roll for the opposition, or because it is not an *opposed test* and the game master will roll 5 dice because the player should not be sure of the outcome. Only in cases where the player is managing the situation fully they should roll the full 10 dice, but either roll 5 dice twice or use differently coloured dice to calculate *Criticals* and other functionality the dice roll is covering.

Every test requires 10 dice to be rolled in total.

In this rule set, 5 FUDGE dice will always be referred to as:

5f

while the full 10 FUDGE dice will always be referred to as:

10f

1.2.1 Result

The Result of *10f* is calculated by rolling 2 times 5 dice and summing all "+" as 1 and all "-" as -1 while blanks count as 0.

If the Result of a *10f* roll needs to be calculated in this rule system it will be denoted as:

10fR

Probability Distribution The average *Result* of any dice roll in Pink Trenchcoat is always 0. The number of total dice rolled is also always 10 (although, sometimes, the

dice are rolled by different people for psychological reasons, mathematically this makes no difference).

Using 10 dice, the following statistics apply the outcome of *10fR*.

Probability for exactly rolling a value Sometimes it is good to know what the probabilities to exactly roll a value are. The probability distribution of the *10fR* is a gaussian with mean of 0 and a standard deviation of about 2.6.

Figure 1.1: *10fR* Probability Distribution

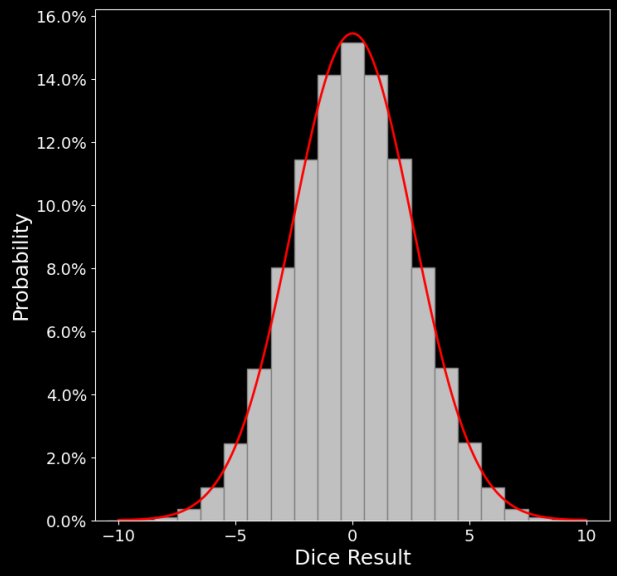

Table 1.1: 10fR Probabilities

Roll exactly	Chance	one in
-10/10	0.0014%	71000
-9/9	0.016%	6100
-8/8	0.088%	1100
-7/7	0.36%	280
-6/6	1.0%	96
-5/5	2.4%	41
-4/4	4.8%	21
-3/3	8.0%	13
-2/2	12%	8.7
-1/1	14%	7.1
0	15%	6.6

Probability for rolling a value and lower/higher Most of the time it is important to know the probability to at at least a certain number or higher, or the inverse, the chance to roll a certain number or lower. Both are important to judge if a *Test* will fail or succeed.

As a rule of thumb, rolling below -5 or above 5 is not happening often. This also means that *Tests* that only fail when a value smaller than -5 is rolled should only be done if the success or how well it succeeded or failed is critical for the game. Instead it can just be assumed that the *Test* succeeded normally.

Figure 1.2: *10fR* Cumulative Probability Distribution

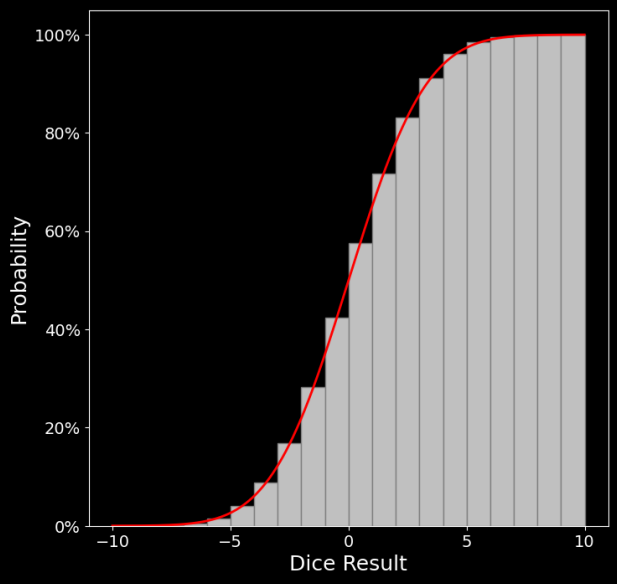

Table 1.2: 10fR Cumulative Probabilities

Roll exactly or		Chance	one in
bigger	smaller		
10	-10	0.0014%	71000
9	-9	0.08%	5600
8	-8	0.11%	940
7	-7	0.46%	220
6	-6	1.5&	66
5	-5	3.9%	25
4	-4	8.8%	11
3	-3	17%	6.0
2	-2	28%	3.5
1	-1	42%	2.4
0	0	58%	1.7

1.2.2 Anomalies and Criticals

The *Result* is not the only quantity that the dice deliver. Another one is Anomalies and Criticals. They are in principle the same thing, but Criticals are much more seldom and extreme in their effect.

Criticals and Anomalies are determined only looking at the *5f* roll of either the player and the game master. This means that both parties in an *Opposed Test* can generate a Critical or Anomaly at the same time. They happen if multiple dice show similar symbols.

Anomaly To determine Anomalies the number of similar symbols have to be counted. Every time 4 dice of a *5f* roll show the same symbol, an Anomaly happened. This can be four "+" (Positive Anomaly), four "-" (Negative Anomaly) or four blanks (Neutral Anomaly).

The chance to roll an Anomaly is 4.1% for any kind of Anomaly. This means that the chance is 12.3% to have any kind of Anomaly in a *Test*. The Game Master needs to decide whether they want to ignore Anomalies in an *Opposing Test*, if the opposing faction is an NPC. The same applies for the other *5f* that are rolled in a *Unopposed Test*.

Positive and Negative Anomaly The result of a positive or negative Anomaly enhances the outcome of the *Test* in a positive or negative way respectively, but does not change the *Result*. The Game Master needs to look at the situation and think of any positive or negative effects that could happen.

This includes:

- Taking more/less time of an action in combat that normally can not be slowed/sped up
- getting into a advantageous/disadvantageous position when performing a melee attack
- increasing/decreasing connection status of a contact when doing legwork
- using less/more resources when crafting an item

Neutral Anomaly A neutral Moderate Critical should just create unusual side effects to an outcome. Again the Game Master should be free to invent anything coming to their mind.

For example:

- A
- b
- c

Critical Criticals happens if all 5 dice of a *5f* show the same symbol. As with Anomalies there are positive, negative and neutral Criticals. Both the chance and the effect of a Critical are much more radical than an Anomaly.

The chance to roll any kind of Critical is 0.4%.

Positive Critical If there is a remote chance of the *Test* succeeding, it will. This does not allow *PC* to do things that are impossible like surviving an atomic blast or succeeding in a wrestling match with a dragon, but anything close to that.

Negative Critical The *Test* fails and it fails spectacularly. The Game Master is free to invent any convenient explanations. There is always a way something can fail.

Neutral Critical The *Result* of the *Test* is not affected, but something very strange happens. The Game Master can do whatever they see fit.

1.2.3 Non Blanks

The Non Blanks of *5f* is calculated by counting all the "+" and "-" symbols, resulting in a number from 0 to 5.

If the Non Blanks need to be calculated from a *Test* this is denoted as:

5fN

Note that does not mean that an additional *5f* need to be rolled in addition to the *10f* of the *Test* itself, but instead use the *5f* from the existing *10f* roll.

The Non Blanks are used for various secondary purposes of a dice roll.

Figure 1.3: *5fN* Probability Distribution

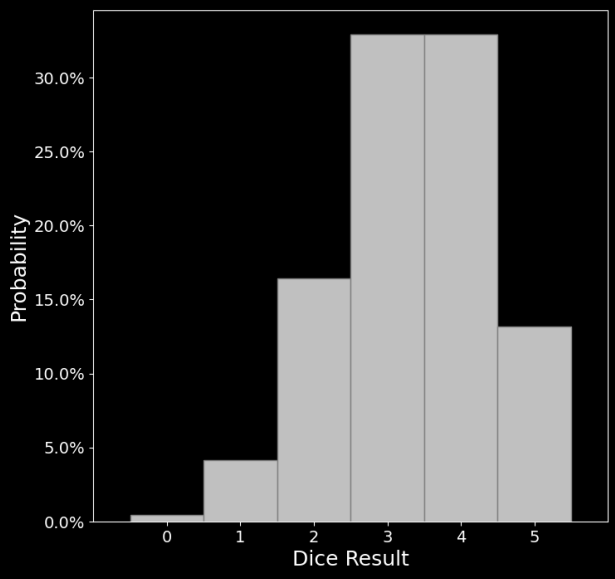

Table 1.3: 5dN Probabilities

Roll exactly	Chance	one in
5	13%	7.6
4	33%	3.0
3	33%	3.0
2	16%	6.1
1	4.1%	24
0	0.4%	240

1.3 Tests

A test determines the outcome of a certain action, which has a certain probability to fail and which has an important impact on the game session if it fails. Tests should not be rolled if it is clear that the test will succeed, like in the case of opening a door. Tests should also not be rolled if the result is irrelevant for the game session, like when a character is trying to beat a popular game in their spare time.

Every time the outcome of an action is, given the capabilities of the acting character, in doubt, or if the result needs to be quantified, a *Test* is rolled.

1.3.1 Test Anatomy

All Tests in Pink Trenchcoat look like the following:

$$\begin{aligned} \textit{Test Quality} = \textit{10fR} + \textit{Ability Score(s)} \\ + \textit{Modifiers(s)} \end{aligned} \tag{1.1}$$

The *10fR* was already explained in the previous section.

Ability Score The Ability Score is a number giving the proficiency of the person or entity that performs the *Test* to achieve the result. The higher, the better.

Normally Ability Scores are either *Attributes* or *Skills* of a character.

Limits Sometimes tools and other situational effects are not modeled as a *Modifier* that is added or subtracted but as a *Limit* to the *Ability Score*. In case the *Ability Score* can not be higher than the *Limit*.

Limits to the *Ability Score* are noted as follows:

$$Ability\ Score(Limit) \tag{1.2}$$

Modifier *Modifier* can be anything from a threshold that needs to be achieved to circumstantial *Modifiers* like visual conditions, tools or wounds that can change the result of a *Test*. If a *Modifier* is helping the *Character* performing the *Test*, like good tools, or support from friends, it is positive. If it is an obstacle of problem for the *Character* performing the *Test*, the *Modifier* is negative.

Test Quality The *Test Quality* ot *TQ* is the value that results from adding the *10fR* the *Ability Score* and the *Modifiers*. If the *Test Quality* is zero or positive, the *Test* succeed, if it negative it failed. The higher the *Test Quality* the better the result and the lower the *Test Quality* the worse the failure.

Table 1.4: Test Quality

TQ	Description
$<$ -9	Epic Fail
-7 to -9	Severe Failure
-4 to -6	Decisive Failure
-1 to -3	Failure
0	Barely made it
1 to 3	Acceptable
4 to 6	Good Result
7 to 9	Exceptional
$>$ 9	Epic Success

1.3.2 Unopposed Tests

In an *Unopposed Tests* a *Character* is not testing against another *Character* but against the environment. Typical *Unopposed Tests* include:

- crafting something
- climbing a wall
- running fast
- remembering something

In this case, the *Ability Score* is just the relevant value from the *Character* and the *Modifier* is the difficulty of the task plus any additional situational *Modifiers*.

This rule system defines the *Ability Scores* to use in an *Unopposed Test* in the following notation:

$$Ability\ Score_{Acting\ Character} + Modifier \tag{1.3}$$

In case of a climbing test for a given wall, that would be:

$$Climbing - 6$$

1.3.3 Opposed Tests

If two *Characters* are fighting against each other, either literally in melee combat or figuratively when one *Character* tries to sneak by and the other to spot the sneaker, an *Opposed Test* is called for. In this case, both involved *Characters Ability Scores* are used. The definition of the *Test* explains which values of a *Character* are used, as this can be the same, in the case of melee combat or be different in the case of sneaking.

This rule system defines the *Ability Scores* to use in an *Opposed Test* in the following notation:

$$Ability\ Score_{Attacker}\ vs.\ Ability\ Score_{Defender} \tag{1.4}$$

In case of melee combat this would mean:

$$Melee\ Combat\ vs.\ Melee\ Combat$$

In case of sneaking it would mean:

$$Stealth\ vs.\ Perception$$

The final *Test Quality* is then calculated as follows:

$$\begin{aligned} Test\ Quality = 10fR \\ &+ Ability\ Score_{Attacker} \\ &+ Modifiers(s)_{Attacker} \\ &- Ability\ Score_{Defender} \\ &- Modifiers(s)_{Defender} \end{aligned} \tag{1.5}$$

1.3.4 Supported Tests

If one or more *Characters* are helping another *Character* to do a task that can not be split into subtasks, but all characters have to do the full task, this is a *Supported Test*.

- climbing a wall together
- helping a character to sneak
- crossing a mine-field

In this case the *Ability Score* for the *Supported Test* is the average *Ability Score* of all the *Characters* involved. The *Modifiers* for the *Supported Test* are the average *Modifiers* of all the *Characters* involved -1.

The *Game Master* decides which *Tests* can be supported.

1.3.5 Collaborative Tests

If one or more *Characters* are working together, distributing the work to perform a task that can be broken down into independent parts this is a *Team-Play Test*. The goal is to either increase the quality of the result, or to speed up the process by using less *Task Time*.

- crafting an item
- collecting information
- repairing a vehicle
- summoning a spirit

In this case the *Ability Score* for the *CollaborativeTest* is the average *Ability Score* of all the *Characters* involved. The *Modifiers* for the *Collaborative Test* are the average *Modifiers* of all the *Characters* with an additional benefit depending on the number of *Characters* working in the *Test*.

Table 1.5: Collaborative Test

Characters	Modifier
3	+1
10	+2
100	+3
1000	+4

The *Game Master* decides which *Tests* can be *Collaborative Tests*.

1.3.6 Task Time

In most *Tests* a *Character* can spend more ore less *Task Time* to do the task better or achieve an outcome faster. In the case of spending more *Task Time*, this will either make a success possible or allow for a better result.

Table 1.6: Extra Time

Time Multiplier	Modifier
x0.5	-6
x0.7	-3
x3	+1
x10	+2
x100	+3
x1000	+4

If not explicitly allowed or disallowed by the rules the *Game Master* decides whether spending more or less *Task Time* is possible.

Chapter 2

Character

This chapter describes *Characters*. Currently this chapter describes only meta-human *Characters* with a physical body to be played by a *Player*. In principle, certain types of *Agents* and *Spirits* could also be played, but are currently not in scope of this rule-set. body in particular.

2.1 Attributes

Attributes are very central values in defining a *Character's* abilities. They give a broad description of a *Character's* strengths and weaknesses and are influencing both final *Skill* values as well as derived *Characteristics*.

The base value for most *Attributes* of an average human is 8.

Table 2.1: Attribute Values

Value	Description
< 4	Disabled
4-5	Challenged
6-7	Underdeveloped
8	Average
9-10	Improved
11-12	Superior
13-14	Exemplar
> 14	Superhuman

Attribute values in Pink Trenchcoat are logarithmic with a base of 3. This means that a *Character* with *Strength* 11 is twice as strong as a *Character* with *Strength* 8 which in turn is twice as strong as a *Character* with *Strength* 5. This fact is only influencing certain *Characteristics* like *Carrying Capacity* and does not need to be kept in mind in most situations.

2.1.1 Mental Attributes

Pink Trenchcoat uses four *Mental Attributes*.

Charisma *Charisma* describes a *Character's* ability to positively affect other people in interactions. Highly charismatic people instantly get the attention of others, are often favored, and respected. A person with a low *Charisma* value is often ignored and sometimes not taken seriously. *Charisma* is also required to connect with people emotionally and understand emotional context of a conversation.

Ware is negatively affecting *Charisma* as it detaches the *Character* from itself.

Inutition *Intuition* describes the *Character's* ability to intuitively ans subconsciously process information. It describes not how fast or how much the *Character* can process, but how well. Furthermore a high *Intuition* value helps the *Character* to grasp a situation faster and perceive better.

Logic *Logic* describes the raw processing power and storage capacity of a *Character's* brain. Combined with *Intuition*, both attributes form the *Character's* IQ. A high *Logic* value helps with most *Craftsmanship* and all *Knowledge Skills*.

Willpower *Willpower* represents the amount of control the *Character* has about their mind and body. How far they can force their body to go, and how well to withstand temptations of any kind. It is also a measure for courage.

2.1.2 Physical Attributes

Pink Trenchcoat uses four *Physical Attributes*.

Agility *Agility* represents a *Character's* nimbleness and dexterity. The motions of a *Character* with high *Agility* look fluid and smooth, while low *Agility* motions look stocky. *Agility* is important for all *Close Combat* and most *Physical Skills*. Larger *Characters* normally have lower *Agility*.

Body Body describes a *Character's* ability to endure physical strain, and keep going, even when exhausted. It also influences how much *Damage* the body can take before collapsing. *Body* is independent of *Size*, meaning that a large *Character* does have the same average *Body* as a smaller one.

Coordination *Coordination* is the ability to control your body the way you want, especially hand-eye coordination. Although a *Character's* body can be very agile, as long as the *character* can not control it in the right way, it may not help much. *Coordination* is important where the *Character* works with his hands, like in *Ranged Combat* or most *Craftsmanship Skills*.

Strength *Strength* measures the raw power of a *Character's* body, the pure muscle volume. Most *Physical Skills* benefit from a high *Strength* value. *Strength* generally increases with *Size*.

2.1.3 Other Attributes

Fate *Fate* is a measure of a *Character's* luck, the favour of the gods or their balance score with the universe itself. Or it is just a gamistic resource that can affect *Tests*.

Fate refreshes every game session and can be used to modify *Test* either before or after the roll. It can only be taken in high stake moments, that are critical for the story or the *Character*. The *Game Master* decides if this is the case.

Optional: Anomaly Ownership A *Player* can also, by spending *Fate*, take ownership of an *Anomaly* they have rolled. This means that now the *Player* instead of the *Game Master* decides and describes what the special effects of the *Anomaly* are. The *Game Master* however needs to accept the effect and decides how much *Fate* it costs.

Table 2.2: Fate Costs

Value	Description
1	+1 before
3	+2 before
6	+3 before
4	+1 after
9	+2 after
1+	own *Anomaly*

Magic

Size

2.2 Characteristics

2.3 Health

Life

Wound Limit

Damage Pip

Wound Heal Time

2.4 Athletics

Carrying Capacity

Combat Speed

Action Costs

Reaction

2.5 Skills

2.5.1 Combat

2.5.2 Physical

2.5.3 Processing

2.5.4 Empathy

2.5.5 Craftsmanship

2.5.6 Resistance

2.5.7 Piloting

2.5.8 Magic

Chapter 3

Combat

Chapter 4

Computers

This chapter explains both the matrix, including AR and everything computer related like electronic warfare.

4.1 What is the Matrix

The Matrix is a virtual representation of the cyberspace for human users. It is they way they perceive interactions between themselves and both other matrix users and *Matrix Entities*.

4.1.1 Accessing the Matrix

There are various ways to access the matrix.

Physical Access This method of matrix access uses outdated methods like keyboard and mouse. It is generally outdated and very slow. It is only used if people are afraid of any kind of matrix damage, or are very traditional.

Augmented Reality Augmented Reality or AR access is a widely used for of matrix access, especially one the go or while wanting to do things in parallel. AR users still see the real world, but get additional information projected on top of it. Thus they can see objects, additional information and also sound added to the real world that does not exist.

Virtual Reality Virtual Reality supersedes the perception of the user. They are not aware of the real world, but instead see, hear, smell and feal virtual sensory input that is 100% artificial.

Tortoise Tortoise uses not direct brain interfaces as provided by most data jacks, but uses outdated technologies like trodes. Due to it not requiring cyberware it is often used by adepts or magicians.

Cold Sim Cold Sim is the standard way of using the matrix today. The user is experiencing the matrix by direct stimulation of their sensory cortex so that they see, hear and feel the matrix. Their thoughts of movements and actions are translated into commands of their virtual bodies using virtual applications.

Hot Sim Hot Sim is the most dangerous but also the fastest way to access the matrix. The data is directly fed into the users brain even circumventing their sensory centers that are stimulated in cold sim. Instead, using knowledge link technology, the matrix user just instantly knows the information. Also their raw thoughts are transformed into matrix commands.

Table 4.1: Matrix Access Methods

Method	Input	Output
Physical	• Keyboard • Mouse • Touchscreen • Input Trigger	• Screen • Loudspeaker
AR	• Transducer • Microphone • AR Gloves • Holo Scanner	• Lenses • Vision-Link • In-Ears • Sound-Link
Tortoise	• Trodes • External Sim Rig	• Trodes • External Sim Module
Cold Sim	• Sim Rig	• Sim Module
Hot Sim	• Transcriber	• Knowledge Link

Table 4.2: Matrix Access Requirements

Method	Processor/ Uplink
Physical	1
AR	3
Tortoise	6
Cold Sim	6
Hot Sim	10

4.2 Matrix building blocks

4.2.1 Matrix Devices

The Matrix is made up of hardware that is processing and delivering it. Most notable are are the different pieces of hardware the matrix is running on. In general four different classes of matrix hardware can be found.

Gadget Gadgets are small and cheap pieces of hardware. Some of them are so cheap, they can be found in throwaway

Table 4.3: Matrix Access Modifiers

Method	Skill	React	Tick	Damage
Physical	-3	-5	x6	None
AR	-2	-3	x3	Fatigue
Tortoise	-1	-2	x1.5	Fatigue
Cold Sim	0	0	x1	Stun
Hot Sim	+2	+3	x0.7	Physical

articles like food packaging. Others are powering small sensors or track positions. They range from pinhead size to coin size. A typical person is carrying around dozens of them.

Commlink Commlinks are not only the most common mans to communicate but also a matrix hardware class. They are bigger than gadgets, but the smallest of them can fit into a bigger earring. The standard size is of an average playing card. They carry enough processing power to allow for at least *Augmented Reality*.

Cyberdeck Cyberdecks are a special form factor that only few people need. Much bigger than a an average commlink, about the size of a shoe-box, they pack much more processing power. Most cyberdecks are used for illegal purposes and are equipped with a *Sleaze* module to avoid detection in the matrix.

Mainframe Mainframes are stationary pieces of matrix hardware. They range from shoe-box size to whole floors of a building. Mainframes are used to service multiple people or perform high performance computations.

4.2.2 Matrix Entities

Matrix entities are virtual building blocks of the matrix. Although they have a physical basis, they are purely virtual representations both in virtual- and augmented reality.

Node A Node is a matrix entity with processing power. It has matrix location and can be *accessed*. A Node can run *Processes*, store *Files* and be the origin or destination of a *Stream*.

Process Processes are matrix entities that actively perform actions. They are running on their origin *Node*.

Persona A Persona is a special kind of *Process* that represents a matrix user and their actions. *Personae* can *access Nodes*. In this case they are connected to their *origin Node* via a *Stream*.

Program A program is a piece of software that can be used by a *Persona* or an *Agent* as a tool to perform various actions. Programs are always attached to a *Persona* or *Agent*.

Agent An agent is a process that can perform autonomous decisions and use *Programs* to perform actions. *Agents* can *access Nodes*. In this case they are connected to their *origin Node* via a *Stream*.

ICE ICE, or Intrusion Countermeasures, are *Agents* with the special purpose to defend a node from hackers.

Streams A stream connects two *Nodes*, the origin and the destination, with a data connection. A stream also connects the *Node* a *Persona* or *Agent* is running on with the *Node* it is *accessing*.

File A *File* is a coherent set of any kind of data.
This includes:

- a text document
- a trideo clip
- a BTL movie
- a voice record

4.2.3 Access Levels

In Pink Trenchcoat a decker that is *accessing* a *Node* is identified with a given *Access Level*, or *Account*. This *Account* is specific to the *Node* and linked to the deckers SIN or, in the case of *Agents*, to their *AID*.

Anonymous

User

Security

Admin

4.2.4 Matrix Properties

Access Rights

Access ID

Subscription List

Logs The *Logs* are a special *File* that contains a history of all actions in a Node, including all actions of *Personae* and *Agents*, their *AIDs*, the *Files* and *Streams* the created and consumed and anything else that was done in the *Node*. *Actions* from a *Process* that has a *Sleaze* rating are only *logged* when they have been successfully *analyzed* by *Analyze ICE*.

4.2.5 Matrix Attributes

Each *matrix device* has a number of attributes that define its properties in the matrix.

Processor The *Processor* attribute represents a *Nodes* row computing power. As most devices are very advanced, a high *Processor* rating is not needed for most every day tasks. High *Processor* ratings are required for intensive tasks like processing Sim-Sense signals for example when using *Cold Sim* or the even more complex *Hot Sim*. The attribute is also useful if a mainframe is supporting a large user base.

It is also important in matrix combat where combatants try to overwhelm the opponents *Node*.

The *Processor* attribute is mostly related to a *Devices* size. The bigger a *Device* the higher its rating is on average.

Table 4.4: Processor Ratings

Entity	Processor
Gadget	0-4
Commlink	3-8
Cyberdeck	6-13
Mainframe	8-21

System System describes the quality of the operating system and standard software suite of a *Node*. The higher the ranking the higher the rating of *Programs* that can be run.

A high Systems rating also helps autonomous software like *ICE* to perform more efficiently.

Firewall Firewall represents the resilience of a *Node* against anything illegal. This includes any kind of *Exploit* actions leading to illegal actions not governed by the users level.

Firewall is not determined by a *Nodes* computing power but by the skill and time invested by the maintainers of the node, and the number of users and different *Processes* it is supporting.

Firewall Ratings are often given by a color coding.

Table 4.5: Firewall Ratings

Color	Firewall
Blue	0-4
Green	5-9
Orange	10-14
Red	15-19
Ultra Violet	20-21

Blue Blue *Nodes* represent the lowest level of security. They are often either very cheap gadgets like Smart Tags or public mainframes like public libraries.

Green Green *Nodes* represent the vast majority of matrix hosts. They are a good trade-off between expensive security experts and time invest. *Nodes* with fewer users tend to have higher green ratings.

Orange Orange *Nodes* are used when higher security is required, like in the mainframe of a police station, a law firm, or the *Nodes* of upper class individuals.

Red Red *Nodes* are mostly used by high security facilities like corporate research sites or government agencies.

Ultra Violet Ultra Violet *Nodes*, if they exist, are only used for legendary and top-secret institutions.

Uplink Uplink describes the quality, speed and volume of data that a *Node* can access per time. A high throughput is required for *Cold Sim* and even more for *Hot Sim*. Uplink mostly degrades over distance, although not as fast as wireless *Signal* does, or if the signal has to go through wireless channels.

Signal The Signal rating describes the power and quality of a wireless signal. It is used to check how far a signal penetrates and also represents the power delivered in case of *Electronic Warfare*. Only nodes with wireless capabilities have a Signal rating.

Table 4.6: Signal Ranges

Signal	Range	Signal	Range
0	1 m	**11**	5 km
1	2 m	**12**	10 km
2	5 m	**13**	20 km
3	10 m	**14**	50 km
4	20 m	**15**	100 km
5	50 m	**16**	200 km
6	100 m	**17**	500 km
7	200 m	**18**	1,000 km
8	500 m	**19**	2,000 km
9	1 km	**20**	5,000 km
10	2 km	**21**	10,000 km

Sleaze Only devices equipped with with an illegal *Sleaze* module have a *Sleaze* rating. The *Sleaze* rating allows a decker to hide from security software of a *Node*. Without it the decker would instantly be recognized after performing any kind of *Exploit* action.

A *Sleaze* module allows also to broadcast and change (fake) SINs the decker possesses. The decker can not mimic arbitrary SINs.

4.3 Matrix Actions

4.3.1 Basic Actions

Basic Actions are very simple and normally do not require a *Test* or *Program*. If a *Test* is required because the *Character* is wounded or has an extreme non-technical background use:

$$Computers + 3$$

Table 4.7: Matrix Actions

Account Level	Program	Node	Process	Stream	File
Anonymous	None	• Anonymous Access			
	Analyze	• Analyze	• Analyze	• Analyze	• Analyze
	Break			• Break	• Break
	Corrupt	• Crash • Slow	• Crash • Slow	• Corrupt	• Corrupt
	Find	• Find	• Find	• Find	• Find
User	None	• User Access	• Command • Start • Stop	• Decrypt • Read • Start • Send • Terminate	• Create • Decrypt • Delete • Read • Write
	Control		• Control [Thing]		
	Crypt			• Encrypt	• Encrypt
	Generate			• Generate	• Generate
	Medic	• Repair	• Repair		
Security	None	• Security Access • View Accounts • View Alert Status • View Logs • View Subscriptions	• Command ICE • Start ICE • Stop ICE		
Admin	None	• Admin Access • Change Alert Status • Edit Accounts • Edit Logs • Edit Subscriptions • Shutdown			

Access Node
Prerequisite *Node AID*
Duration 0.1s

This action is required to access a *Node* with a known *AID*. After a successful *Access Action* the decker has *accessed* the *Node*.

Having *accessed* a *Node* is often a prerequisite for lots of *Matrix Actions* targeting *Files* and *Streams*. It is only of particular relevance when a decker does not have the relevant *Access Rights* to *access* the *Node* and needs to *Exploit* their way in.

Change Alert Status
Prerequisite *Accessed Node*
Duration 0.5s

This action allows the decker to change the *Nodes Alert Status*.

Command
Prerequisite *Process AID*, *Accessed origin Node*
Duration 2s

This action allows a decker to give commands to a *Process*. This can either be an *Agent*, or any other *Program* on a *Node* or *Device* like a drone or a security camera.

The decker needs the *AID* of the *Process* and needs to *access* the origin or target *Node* of the *Process*.

Create File
Prerequisite *Accessed Node*
Duration 1s

This action creates a *File* in a *Node*. The creator chooses content and *Access Rights* and gets the *Files AID*.

Decrypt
Prerequisite *Red File*, *CryptKey*
Duration 0.1s

Decrypt and *encrypted File* if the decker has the *CryptKey*.

Delete File
Prerequisite *File AID*, *Accessed Node*
Duration 0.1s

Delete a *File* in a *Node*. After the *File* is *deleted* it can not be recovered.

Edit Accounts
Prerequisite *Accessed Node*
Duration 0.5s

This action allows the decker to edit *Accounts* of a *Nodes*. This includes removing, adding and changing *Access Levels*. In the case of adding a new *Accounts* the respective SIN is required.

Edit Logs
Prerequisite *Accessed Node*
Duration 0.5s

This action allows the decker to edit the *Logs* of a *Node*. This includes adding and removing entries.

Edit Subscriptions
Prerequisite *Accessed Node*
Admin Access other Node
Duration 0.5s

This action allows the decker to edit the *Subscription List* of a *Nodes*. This includes removing and adding *Nodes*. In the case of adding a decker needs Admin Access on the other *Node*.

Read File
Prerequisite *File AID*, *Accessed Node*
Duration 0.1s

This action allows a decker to read *Files* in a *Node*. *Reading* a *File* enables a decker to *create* a local *File* copy in the *Personas* origin *Node*.

Read Stream
Prerequisite *Stream AID*
Accessed origin/target Node
Duration 0.1s

This action allows a decker to read *Streams* in a *Node*. *Reading* a *Stream* enables a decker to *create* a local *File* containing the content of the *Stream* in the *Personas* origin *Node*.

Start Process
Prerequisite *Accessed Node*
Duration 1s

This action creates a *Process* in a *Node*. The creator chooses its *Access Rights* and gets the *Process AID*.

Send to Stream
Prerequisite *Stream AID*
Accessed origin Node
Duration 1s

This action creates a *Stream* between two *Nodes*. The creator chooses content and *Access Rights*.

Start Stream
Prerequisite *Accessed origin Node*
Accessed destination Node
Duration 1s

This action creates a *Stream* between two *Nodes*. The creator chooses content and *Access Rights* and gets the *Streams AID*.

Stop Process
Prerequisite *Process AID*
Accessed origin Node
Duration 1s

This action *stops* a *Process*. A related *Agent* or *Persona* is instantly shut down.

Terminate Stream
Prerequisite *Stream AID*
Accessed origin Node
Duration 0.5s

This action *terminates* a *Stream*. A related *Process* is instantly stopped.

View Accounts
Prerequisite *Accessed Node*
Duration 0.5s

This action allows the decker to view all *User*, *Security* and *Admin Accounts* for the *Node*.

View Alert Status
Prerequisite *Accessed Node*
Duration 0.5s

This action allows the decker to view the current *Alert Status* of the *Node*.

View Logs
Prerequisite *Accessed Node*
Duration 0.5s

This action allows the decker to view the current *Logs* of the *Node*.

View Subscriptions
Prerequisite *Accessed Node*
Duration 0.5s

This action allows the decker to view the *AIDs* of the *Nodes* the are *subscribed* to the *Node*.

Write to File
Prerequisite Found *File*
Duration 0.5s

This action allows a decker to *write* any content to a *File*.

4.3.2 Advanced Actions

Advanced Actions require *Tests* to perform and require a *Program* to carry out. The standard *Test* is:

$$Computers(Program) + Modifiers$$

Analyze [Node, Process, Stream, File]

Program	Computer(Analyze)
Prerequisite	Found [Node, Process, Stream, File]
Test Modifier	Target *Sleaze*
Duration	2s

This action allows for analyzing properties of various matrix entities. To analyze a *Node* an AID is required. Other entities have to be *found*. *Processes* and *Streams* can only be analyzed if the decker has *accessed* either the target or the destination *Node*.

Table 4.8: Analyze Node Results

Result	Properties	Location
0	Active Alert Status	
2	AID	
4	Type	
6	High/Low Attributes	Continent
8	Functionality	State
10	High/Med/Low Attributes	City
12	Active Processes	Suburb
14	Exact Attributes	Street
16		Building
18		Room
20		Exact

Control

Program	Skill(Control)
Prerequisite	*Accessed Node* *Process AID*
Test Modifier	var.
Duration	1s

Using the *Control Action* the decker can use any kind of item that can be *controlled* remotely from a *Process* in a *Node*. The decker has to use the relevant *Skill* limited by the *Control Program*.

Skill(Control)

Examples are using remotely controlled guns using *Gunnery* or driving a remotely controlled car using *Wheeled*.

Encrypt

Program	Crypt
Prerequisite	*Accessed Node* *File AID* or *Stream AID*
Test Modifier	None
Duration	1s

The *Encrypt Action* encrypts a *File* or *Stream* so that even deckers with the required *Access Rights* can not read the content. The *Action* does not require a *Test* but automatically encrypts the *File* or *Stream* with the *Crypt Programs* rating. To read the content one needs either the key or try to *Break* the encryption.

Find Process

Program	Computer(Find)
Prerequisite	*Access* to origin/destination *Node*
Test Modifier	*Sleaze*
Duration	10s

This action allows a decker to find *Processes* in a *Node*, which must be either its origin or the destination.

Find Stream

Program	Computer(Find)
Prerequisite	*Access* to origin/destination *Node*
Test Modifier	*Sleaze*
Duration	10s

This action allows a decker to find *Streams* in a *Node*, which must be either its origin or the destination.

Find File

Program	Computer(Find)
Prerequisite	*Access* to origin/destination *Node*
Test Modifier	*Sleaze*
Duration	10s

This action allows a decker to find *Files* in a *Node*.

4.3.3 Matrix Combat

Corrupt

Program	Corrupt
Prerequisite	*Accessed Node* *File AID* or *Stream AID* or Found [*File*/*Stream*]
Test Modifier	Originating *Node* System
Duration	1s

If a decker does not have the *Access Right* to *delete* a *File* or *Terminate* a *Stream* the decker can *corrupt* it so it becomes unusable.

TQ = Cyber Combat(Corrupt) - System

For each point of *Test Quality* deal *Processor Matrix Damage* to the *File* or *Stream*.

Crash

Program	Cyber Combat(Crash)
Prerequisite	Found [*Node*/*Process*]
Test Modifier	System
Duration	1s

For each point of *Test Quality* deal *Processor Matrix Damage* to the *Node* or *Process*.

Repair

Program	Computer(Medic)
Prerequisite	AID
Test Modifier	None
Duration	10s

The *Repair Action* allows a decker to *repair Matrix Damage* on *Nodes*, *Processes*, *Files* and *Streams*. For each point of *Test Quality* repairs one point of *Matrix Damage* to the target.

Slow Node

Program	Hacking(Slow)
Prerequisite	*Access* to *Node*
Test Modifier	System
Duration	1s

The *Slow Action* allows a decker to reduce the *Processor* a target *Node*. Reduce the *Processor* by the *Test Quality*. This effect lasts for 10s, or till the decker takes another *Action*, whichever happens later.

Slow Process

Program	Hacking(Slow)
Prerequisite	Found *Process*
Test Modifier	System
Duration	1s

The *Slow Action* allows a decker to force *Actions* of a target *Node* or *Process* to take longer.

4.3.4 Cracking

Break

Program	Break
Prerequisite	Found [*File*/*Stream*]
Test Modifier	Crypt Rating +3
Duration	20s

Exploit Every time a decker wants to perform an action where their *Access Level* is not high enough, like *Viewing* the security *Log* without being at least *Security* level, an *Exploit Test* is required. If the *Action* in question requires a *Test* itself, when for example *Writing to a Stream*, the *Exploit* test does not replace the actual *Test* but is an additional requirement.

An Exploit test is an opposed test between the deckers *Cracking(Exploit)* and the *Nodes Firewall*.

$$Test\ Quality = 10fR + Cracking(Exploit) - Firewall$$

In addition each Exploit test can increase the deckers *Security Tally*.

Table 4.9: Exploit Modifiers

Account Level	Mods	
	Action	Account
User	0	-3
Security	-3	-5
Admin	-4	-6

4.3.5 Related Actions

Physical Reboot

A *Physical Reboot* can only be done while having physical access to the *Node*. It does not require a test and takes time depending on the *Processor* of the *Node*:

$$Reboot\ Time = Processor^2 + seconds$$

During a *Physical Reboot* the Admin Account can be changed to whatever SIN or *AID* the person that does the *Reboot* desires.

Jack Out

Data Search

4.3.6 Security Tally

Table 4.10: Security Tally Measures

Tally	Measure
5	Analyze ICE
10	Trace ICE
15	Silent Alert
20	Combat ICE
25	Active Alert
50	Emergency Shutdown

$$Tally = 5fN + System - Sleaze$$

Analyze ICE Analyze ICE is looking into a deckers activities to find any signs of illegal actions. If it finds anything it will be added to the deckers security tally.

Trace ICE Trace ICE will try to find the deckers location by analyzing its *Stream*.

Passive Alert In silent or passive Alert Status a list of predefined personnel is informed of a possible intrusion. The Node diverts resources to security purposes, increasing Firewall by 2 and decreasing Processor by 2. Any standard functionality of the Node could be impaired by this resource transfer (GM discretion). The information is not broadcasted to Processes in the Node.

Combat ICE Combat ICE will continuously attack the decker till it is crashed and restart afterwards to attack again.

Active Alert In active Alert Status a list of predefined personnel is informed of an intrusion. The Node diverts resources to security purposes, increasing Firewall by 3 and decreasing Processor by 3. Any standard functionality of the Node can be impaired by this resource transfer (GM discretion). The information is broadcasted to all Processes in the Node.

Shutdown

4.4 Electronic Warfare

Find Wireless

Program	Scan
Prerequisite	Target in *Signal* range
Test Modifier	var.
Duration	10s

Jam Wireless

Program	Scan
Prerequisite	None
Test Modifier	0
Duration	None

Chapter 5

Magic

List of Tables

List of Figures

Alphabetical Index

Appendix A

Combat Tables