User manual

TRSign

version 2.0 (February 12, 2012)

english & deutsch

Contents

1 English

1.1 Introduction

TPSign is a teleport device, working grid-wide by using the RLV technology. TP-Sign was once developed as a RLV teleporter. Now, in the version 2, it was migrated to RTS technology. RTS is a shorthand for '*RLV Teleporter System*'. It is a framework for building of RLV teleporters. RTS offers this way a number of features to TPSign.

Why is TPSign different? TPSign is a grid-wide teleporter, It acheives any location that one can visit via a landmark. The avatar is not moved physically, the teleport not fails if the target location is on different sim or continent. TPSign also doesn't open the map window, like map teleporters do. instead, TPSign passes over the target coordinate to the viewer, enforcing the immediate teleport.

Basically, RLV teleporters require the viewer to support a special RLV interface to pass over the target coordinate. TPSign works this way, too. But since not every viewer suports the interface, TPSign works also without it: TPSign allows to switch the RLV support off and turns than to an usual landmark giver.

Also the supplied TP Relay turns any RLV teleporter (also TPSign with active RLV support) into a map teleporter. This allows to use RLV teleporters, TPSign and every map teleporter on same way and with any viewer. For this reason TPSign passes this relay over to users not running RLV.

1.1.1 Audience

Who is TPSign designed for? Well, it allows to jump over to places on the same sim, like a skybox, club or another floor in the house. This are local teleports, and there are also simplier and lighter solutions to acheive that goals.

But if the task is to make any place on the grid accessible by a single click. If in needs of a teleport device for favorite locations or a network of partner teleporters is planned, no matter if the sims are connected or spread over the whole grid, then is TPSign one of devices of choice.

RTS was designed for managing multiple teleport destinations. For traditional reasons, TPSign is a single-target device. But with a little effort, one can extend it by multi-target ability. The package supplies a step-by-step tutorial about how to do this, and also the rebuilt teleporter: TPVendor is a RTS Teleporter which manages up to 80 destinations.

1.1.2 Scope

The documentation structure is this: The *introduction* explains shortly the content of the product package and how to update it. Also, it explains what RLV is, why we need this interface, which viewers support it and, finally, how to use TPSign without running RLV-capable viewer.

The section 1.2 handles the *installation* and application of the teleporter. Its *configuration* is, again, explained in the section 1.3.

The section 1.4 follows, giving a brief introduction in the how the teleporter works and also a stepwise tutorial about how to extend TPSign to TPVendor, a teleporter using the multitarget feature of RTS.

TPSign is a RTS teleporter. Hence, the RTS documentation is important for background information going farther than both last sections. The RTS documentation you can download on the RTS product page[1], it is linked as a single PDF file.

1.1.3 Package content

The package content is shown in the figure 1.1. Among others, the package contains the teleporter itself, as well its extension, *TPVendor*. Both are ready to use, but need a reconfiguration, for instance to define the teleport destinations.

The user manual for TPSign is inside the notecard *TPSign User Manual* and the required RTS documentation is to find inside the *RTS User Manual* notecard. Both include further notes with partial documentation and in german.

Unpacking

There are two ways to unpack the package, traditional and automated. In first case, please rezz it on floor, ignore the menu, open the package and copy the content. You receive than a folder named '**TPSign (BOXED)**'.

[1] https://marketplace.secondlife.com/p/RLV-Teleporter-System-private-edition/2944310

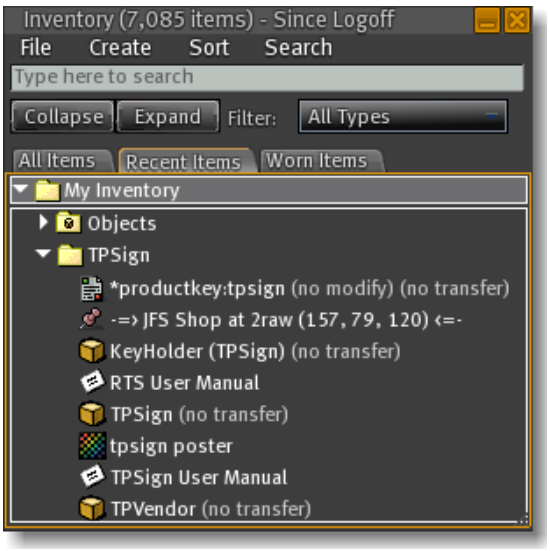

Figure 1.1: TPSign, package content

You can also let the package box open itself. To do so, rezz the box on floor or wear in hand (if rezzing is not allowed.) If you are on a place allowing scripts, a menu opens. If you ignore it, you can reopen the menu by clicking the box.

In this menu, please hit the '**Open**' button. The box will give you the product folder '**TPSign**' and then opens the menu again, where you need the '**Remove**'. This button destroys the box or detaches it if worn.

Please, keep the product package as backup for the files, or at least the script '***productkey**' from it. It is the only way to request product updates manually. There is also an online tutorial about automated unpacking product boxes available.[2]

Update

TPSign is inside a vending system at time, which allows automatic delivery of updates. But you also can request updates manually. To do so, you have two ways: Via the product box itself and via the supplied key holder.

In both cases the script '***productKey**' is working, it connects to an update orb installed in one of JFS stores. The connection is local via chat, hence you need to visit the shop for it. Since they move sometimes, a blog post provides the list.[3]

Instead of the *product box* you can use any prim with installed '***productKey**' script. This is how to request updates by using this way:

[2] http://jennassl.blogspot.com/2011/11/unpacking-jfs-boxes.html
[3] http://jennassl.blogspot.com/2010/05/actual-shop-list.html

Come closer than 10m to the update orb. Now rez please the product box on the ground, a menu opens. Here, hit the '**Update**' button. The update orb will hang out the update, the menu opens again, you can hit the '**Remove**' button there.

The *KeyHolder* is a watch-like device made to keep up to 12 product keys. This way you can update up to 12 products by using a single device. To make the key holder work, you need to install the '***productKey**' script into. In the supplied device the script is already installed. To request updated by using key holder, please do this:

Come closer than 10m to the update orb. Wear the key holder and touch it. A menu opens. The RTS product should be selected. If not, please select it. After a click on the '**Update**' button, the update orb hang out the product update. You can take off the key holder now.

In both cases, the update is a redelivery of the actual product box. There is also an online tutorial about updating UFS products available.[4]

1.1.4 Why RLV teleporters

RLV was a special viewer originally, made with a goal of support the immersion of the user (the feeling, being in the shown scene.) Herefor, the viewer offers a special interface, which allows scripts to gain a limited control over the viewer.

With this interface, scripts can run some of actions on the viewer, usually the user of the viewer can do, like stand up if sitting, or teleport to a position. The list of possible actions is relatively huge, while TPSign only sends the teleport command to the viewer.

If this command arrives the viewer, it connects immediately to the target place. The user sees only the progress bar and short time later the avatar standing on target place. Quite same like by using a landmark or world map, but without opening of landmark or map window.

Exactly here is the difference of RTS teleporters (or in common of RLV teleporter) against the landmark distributors or map teleporters: The user is not pulled out of the scene by opened landmark or world map window.

Originally, only the RLV implemented this interface, but since the developer of the viewer has published it,[5] other creators could make theirs objects RLV-capable and make the viewer more and more popular. Now is this interface also available for all viewer developer as '*RLVa*'[6] and is implemented by many other third party viewers.

[4] http://jennassl.blogspot.com/2011/11/updating-jfs-products.html
[5] http://wiki.secondlife.com/wiki/LSL_Protocol/RestrainedLifeAPI
[6] http://rlva.catznip.com/blog/2009/10/release-rlva-1-0-5/

Without RLV

A disadvantage of RLV teleporters is, not all viewers support the required interface. A user of a viewer which not supports RLV interface cannot use those teleporters in described way. Especially the official viewer doesn't support the RLV interface.

While developing of RTS this was taken in account: RTS teleporters, and hence also TPSign, allow to switch the RLV support off in the configuration. The teleporter works with any viewer than, but instead of accessing the RLV interface just provides a landmark or SLURL to the target place.

With a teleport relay

Another possibility to use RLV/RTS teleporters (hence RTS TPSign) with a viewer without support of the RLV interface, is to use the teleport relay.

To use a RLV teleporter is generally not only a RLV-capable viewer required, but also a special device, the relay. RLV has a built-in technical safety gate: Only RLV commands are accepted which come from an owned device. RLV teleporters belong usually not to the user of the viewer and cannot access the viewer directly.

The relay is an object, worn usually on the avatar body or HUD, hence it belongs to the user of the viewer. It receives commands sent by RLV devices and relies the commands to the viewer, which performs the command. Usually, the relay performs a security check: Is the device allowed to control the viewer or not? In case of doubt asks the relay the user via menu, but if a device is on black or white list of the relay, the relay declines or accepts the commands automatically.

Common relays follow the specification, they require running the RLV-capable viewer and relay all commands to the viewer. But to use RLV teleporters, only three commands are relevant: The *version check*, the *unsit command* and the *teleport command*.

Only this three commands the teleport relay accepts and relies to the viewer and declines every other. This allows to use a RLV teleporter without risk of an unwanted action of RLV devices. By using other relays one has to deal with theirs security settings first.

If a viewer without the RLV interface is used, the teleport relay emulates the interface by sending a faked version response to the teleport device, and by opening the map window for teleport.

This way inworld devices have a limited control over RLV-capable viewer. With a viewer that can't RLV, RLV teleporters become also usable – they turn to a regular map teleporters. The relay was created with this purpose in mind and is installed into TPSign to hang out if the user has no relay in use.

RLV-capable viewers

There are a few viewers meanwhile that support the RLV interface and make it possible to use RLV teleporters with an usual relay. Some of them are listed below.

- Classical RLV (http://www.erestraint.com/realrestraint/)
- Cool Viewer (http://sldev.free.fr/)
- Firestorm and Phoenix viewers (http://www.phoenixviewer.com/)
- Imprudence (http://blog.kokuaviewer.org/)
- Rainbow (http://my.opera.com/boylane/blog/)
- Singularity (http://www.singularityviewer.org/)

The list must not be complete and should give just an overview.

1.1.5 Global coordinates

Every place in SL can be given by a local and global coordinate. Local coordinate defines the position of a point relative to a special given origin.

For example you can see in the symbol bar the local position of the avatar relative to the sim origin. Be it a point with coordinate **<128, 128, 24>** (exactly in the middle the sim.) On the whole SL world, there are as many points with local coordinate **<128, 128, 24>**, as many sims there are. So, only if the sim is specified, you can position a point by a local coordinate clearly in the SL world.

A SLURL gives, hence, both, the local position and the sim which origin is used to define the coordinate. Such a description is distinct in the SL world:

http://slurl.com/secondlife/DaBoom/128/128/24.

All sims join to a *Grid*, the world of SL. Hence, for every sim exists a global, i.e. in the whole grid clear coordinate of the sim origin. So, one can determine that the sim *Da Boom* has the origin on the global position **<256000, 256000, 0>**. How to determine this? For example via the *Grid Survey* service, we just call this URL:

http://api.gridsurvey.com/simquery.php?region=DaBoom

The response contains two numbers: **x=1000**, **y=1000**. We multiple them by 256 (the width of a sim) and assume **z=0** to get the global position.

If we combine the local coordinate of a place in a sim, with the global coordinate of the sim origin, we get the global position of the point in the grid. So, the point given by position '**Da Boom/128/128/24**' is located on the position of **<256128, 256128, 24>**, relative to the grid origin, i.e. global.

The global position is distinct for the whole world of SL, it is also simple (a single vector value.) Map and RLV/RTS teleporters work for this reason with the global position.

The global position has one disadvantage: Sometimes, move sims across the grid. Moving a sim means changing the global coordinate of the sim origin. This also changes the global positions of all places on the sim. Teleporters cannot achieve the places after the move by using old coordinates and must recalculate them.

As a work-around, some teleporters calculate the target position for every teleport again and again. RTS (hence also TRSign) avoid this permanent calculation, which stresses the sim. Instead, the teleporter caches the calculated value and uses it until the calculation was too long ago. Also, the device owner can delete the cached value making the teleporter calculate the position for the next teleport.

1.2 Installation and application

Figure 1.2: Installation

TPSign is a device, using a single prim, formed to a square plane. The installation takes three steps:

1. Rezz the TPSign and align at the workplace, for example a wall, table or a console, figure 2.2. After owner change, the device initializes automatically. This takes a few seconds.
2. Optionally you can configure the teleporter and define the target destination, section 1.3.3.
3. Now you can activate the device as described in section 1.2.1.

The teleporter must not stay a square. You also can form it to a sphere, a button or give it even a sculpted form. In every case, it is a device which reacts on touch and teleports the clicking person to a single target. In the section 1.4 is explained how to extend the teleporter in a way it allows to select the teleport destination.

1.2.1 Taking in operation

The TPSign is ready to use straight out of the box, but it will teleport to the preconfigred destination. If you don't know RTS teleporters, it is advisable to test TPSign as delivered and then change the configuration.

You can control the teleporter via owner menu, which is shown only to the owner. TPSign uses a single prim, there is no extra menu button, to open the owner menu, please use a longclick.[7] The menu has four buttons:

[7]Please click the teleporter and hold the left mouse button one second.

- '**start**', '**stop**' to start or stop the teleporter.
- '**reset**': Resets all scripts and initializes the device. This also reads the configuration and sets up the device. As soon you changed the configuration, or the device, please remember to reset the teleporter.
- '**refresh**': This button removes the cached global coordinate (section 1.1.5) and enforces so the calculation for the next teleport.

1.2.2 Application

To travel with TPSign, a single click on the teleporter is enough. What happens next depends on if it is in RLV or LM mode and also on if the user uses RLV capable viewer and an active relay. The complete process is shown in figure 1.3.

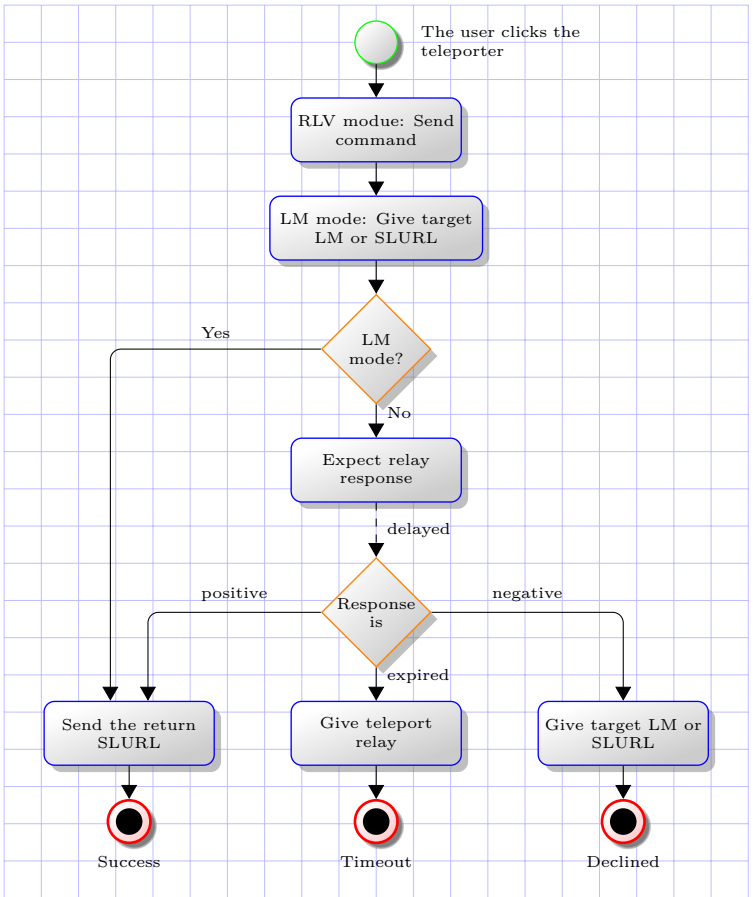

Figure 1.3: Teleport process

In the RLV mode

Is the RLV mode active, the teleporter sends a teleport request naming the user. The request is received by the worn relay owned by the user.

Is the request accepted, the relay passes it over to the viewer. *The viewer performs the teleport.* The teleporter receives the positive response from the relay and (if configured so) gives the SLURL of the own position to the user via IM (the return-SLURL.)

Is the request declined, the teleporter can give the user only the LM or SLURL to the target place, if the LM mode is active.

Is the relay absent, switched off or the user took too long to answer the relay menu, the teleporter detects timeout and hangs out the teleport relay to the user (or another file set up in the configuration.)

In the LM mode

Is the LM mode active, no matter if the RLV mode is active or not, the teleporter gives imediately the target LM or SLURL. The teleport is then valued as successed because the user can use the given LM or SLURL.

Is the RLV mode active at same time, the teleport request is sent via RLV anyway, and processed by the viewer eventually, but the teleporter doesn't await the relay response.

1.3 Configuration

The configuration of TPSign (like the RTS teleporters) is done by a notecard, which is inside the object inventory of the teleporter and has the name '**config**'. If delivered, TPSign is configured by a text shown in 1.4.

```
# TPSign Configuration
[*MAIN]
user         = all

image        = 09BB628B-F97A-44FD-9503-6E4EC62DE59E
side         = all

title        = %n\n...touch to visit
color        = lavender
fadeout      = 0

mode         = RLV
fail         = LM
return       = LM
delay        = 10

norelay      = TP Relay (r upper arm)
offset       = -3
fasttp       = OFF
refresh      = 24

format       = APP
msg:url      = This link brings you at target location:
msg:lm       = This landmark brings you at target location...
msg:return   = This link brings you back:
msg:norelay  = Please, use this item in order to use me...

# The only target
[Welcome to Endora]
image        = 8cee492c-917e-341d-b3a9-63d83ed44d34
target       = Endora/127/169/42
```

Figure 1.4: Configuration of the TPSign

For the detailed configuration info, please read the RTS documentation. Here we only discuss this configuration text and setting up the TPSign specially.

1.3.1 Syntax

The char '**#**' starts the comment: The char and following text until the end of line is ignored. The text before the comment char is processed (except the spaces.) Comments are always closed by the end of line.

Empty lines have no relevance for TPSign, they just improve the readability.[8]

Other lines have either section names or parameter values. A section name is always enclosed in square brakets. Section names are case-insensitive. Around the square brakets are only spaces allowed.

Params are separated from their values by assignment char. Both, param and value must be noted. Params are case-insensitive, some values are also, but not generally.

All param values belong to a last section that was started before the param line. The order in which the params are noted is not relevant.

1.3.2 Main section

```
[*MAIN]
```

The main section has the name '***main**' in any case. Upper case is preferred to emphasize its meaning: this section configures the teleporter generally.

User parameter

```
user        = all
```

The param '**user**' defines who is allowed to use the teleporter, i.e. to travel with it. The value is a list of string flags, connected by '**+**'. Allowed are the flags '**owner**', '**!owner**', '**group**', '**!group**', '**all**' and '**!all**'.

Are the flags '**all**', '**!all**' used, others will be ignored. In the first case everyone can use the teleporter. In the second one – nobody. The flags '**owner**', '**!owner**' allow and forbid the device owner to use the teleporter. The flags '**group**' und '**!group**' allow or forbid every person from the device group to travel with the teleporter.

In the original configuration 1.4 everyone is allowed to use the teleporter, while the value '**group+!owner**' would mean: Everyone off the device group may travel with the teleporter but not the owner.

[8] For performance reasons, please avoid empty lines and pure comment lines.

Picture parameters

```
image       = 09BB628B-F97A-44FD-9503-6E4EC62DE59E
side        = all
```

The param '**image**' defines the default picture. It is used if a selected target not defines an own. Allowed is either the UUID of the picture, or name of the image in object inventory. It must be full-perm there, otherwise its UUID can not be resolved.

The param '**side**' gives the face number which will show the target image. Possible values are '**all**', '**none**' and a number between 0 and 9. The value '**all**' means: the target picture is shown at all faces, and the value '**none**' suppress displaying the picture. If a number is given, this is the number of the face that takes the picture.

Title parameters

```
title       = %n\n...touch to visit
color       = lavender
fadeout     = 0
```

The param '**title**' defines the title (hover text), the value supports placeholders:

- The placeholders '**\n**' and '**\t**' are replaced by the line break and tabulator.
- The placeholders '**%n**' and '**%s**' are replaced by the destination name and name of the defining landmark or target sim respectively.
- The placeholders '**%p**' and '**%c**' are replaced by the number of the selected target in the list (1 means first target) and amount of targets in the list.

To hide the title, just remove the title param, or use '**-**' as value.

The param '**color**' defines the color of shown hover text. The param has also alias '**colour**'. The value is basically a vector like in example above. But some popular names can be used by color name instead of vector. Names, the RTS understands are given in the RTS documentaion. The color name is case-insensitive, '**White**' means the same as '**WHITE**' or '**white**'.

The param '**fadeout**' defines finally, how long the title is visible before it disappears. The time is set in seconds, the timer starts after the title is changed. 0 means the title remains permanently.

Teleport modes

```
mode        = RLV
fail        = LM
return      = LM
delay       = 10
```

The value of params '**mode**', '**fail**' and '**return**' is a combination of string flags '**RLV**', '**LM**', '**N**' and '**OFF**'. The flags are combined via '**+**' and define the behaviour of the teleporter. The flags work additively: The value '**LM+OFF**' means the same as '**LM**' or '**lm**', flag names are case-insensitive.

Is the flag '**RLV**' used, the teleporter access the RLV interface of the viewer. The flag '**LM**' allows handing out the target LM or SLURL, the teleporter becomes a landmark distributor. The flags '**N**' and '**OFF**' deactivate the mode using this flag.

The param '**mode**' defines how the teleport works, if via RLV or by handing out the LM or SLURL. Allowed value flags are '**RLV**', '**LM**', '**N**' and '**OFF**'. If the flag '**LM**' is set, the teleport is taken as succeed, since the agent can use given LM or SLURL any time.

The param '**fail**' defines what to do if the teleport fails: The teleporter can give the target LM or do nothing. Allowed are the values '**LM**', '**N**' and '**OFF**'.

The param '**return**' defines what to do if the teleport succeed: If allowed, the teleporter will send the SLURL for teleporter's position, so the traveller can return. Allowed values are '**LM**', '**N**' and '**OFF**'.

The param '**delay**' defines how long the teleporter has to await the response of the user's relay until teleport failure is detected and the '**fail**' mode is active. The time is given in seconds, minimum is 5. This time covers the delays due lag, as well time the user took to react on their relay, figure 1.3.

Teleport relay

```
norelay     = TP Relay (r upper arm)
```

The param '**norelay**' defines a file to be given if teleport failed because the user has no relay. The file is addressed by 'teleport relay' in the figure 1.3, the timeout branch.

This way the teleporter can give out a notecard with further information. TPSign is configured by giving out the teleport relay, since it allows to use the teleporter even without a RLV-capable viewer.

Offset parameter

```
offset = -3          # means...
offset = <-3, 0, 0>
```

The param '**offset**' gives the return position (rezzing position for returning avatars) relative to the teleporter. You can define it by a single number or by a vector, while the single number equals the vector with that number as x. Both definitions in the example above are equal.

What means this value? If the offset is zero, the return position is the position of the teleporter itself, than avatar using the return SLURL will rez at the position of the teleporter. In some cases it is awkward. If, for instance, the teleporter is mount on a desk or a wall, SL might choose to rezz the avatar under the desk or behind the wall. The offset setting can move this position into a place safe to rezz.

Note: The position of the avatar is the middle of its bounding box, this point is roughly one meter above the ground.

The offset vector is always the vector in the global xy plane around the system prim. If the system prim is not rotated, than the vector `<3, 0, 0>` aims a point 3 meters towards the positive x axis of the system prim. If we rotate the prim around the z axis, the return point follows the rotation. But it ignores the rotations around x or y axis.

This way we can use a teleport sign placed on a wall and configure the offset vector in a way, the return position is 3 meters in front of the sign. Than we can place the teleporter on any other wall and the return position remains in front of it. But if we turn the sign to place is flat on the floor, the return position not moves above the teleporter, it remains 3 meters aside.

FastTP mode

```
fasttp      = OFF
```

This param is included into the TPSign configuration for compatibility reasons. The param value is one of four flags: '`Y`' and '`ON`' switch the mode on and flags '`N`' and '`OFF`' switch the mode off. Since TPSign is a teleporter for a single target, this mode has no relevance.

In a teleporter for multiple targets and with a possibility to select one of them in any way,[9] you can switch the FastTP mode on. If active, the teleport starts immediately as soon a destination is selected in any way. If the mode is inactive, you can scroll through targets and trigger the teleport after you chosen the destination.

[9] TPVendor, described in the section 1.4 is one of such teleporters.

Target refreshing

```
refresh      = 24
```

For teleport, TPSign needs a global coordinate of the target position. Sims move sometimes, which change this value. The global position needs a recalculation and that means delays and server load.

To lower the server load, the resolved coordinate is cached. The param '**refresh**' states how long the saved coordinate remains valid. The value is in hours, the minimum is 6.

The position is not refreshed permanently but due a teleport to an expired position. Thus, it is advisable to use a value of one day to one week here, i.e. 24 to 168 hours. Also, the 'refresh' button in owner menu sets the targets as expired manually.

Message format

```
format = APP
```

The param '**format**' defines the format for sent SLURL. Here you can define one of four values, '**MAPS**', '**SLURL**', '**APP**' and a freetext.

'**SLURL**': Send a *SLURL*.[10] It is classical, all viewers understand the link and open the landmark window if it is clicked in chat history.

'**MAPS**': Send a *MapURL*.[11] It was introduced with the version 1.23 of SL viewer. Older viewers open the web browser instead a landmark window.

APP: Send a *teleport link*.[12]. This string enforces an immediate teleport if clicked in chat history, without to open the landmark window.

A freetext is a text with placeholders `%s`, `%x`, `%y`, `%z`, replaced by the sim name, and local coordinates on the sim, respectivelly. For example the format string '`%s (%x, %y, %z)`' produces a text like 'Endora (123, 234, 345)'.

Note: You can emulate the flags '**MAPS**', '**SLURL**' and '**APP**' by using the freetext format, for example the format '`secondlife:///app/teleport/%s/%x/%y/%z`' produces the same teleport link as for using the '**APP**' flag (but takes more processor time to calculate).

[10] A link like http://slurl.com/secondlife/Endora/123/234/345
[11] A link like http://maps.secondlife.com/secondlife/Endora/123/234/345
[12] A string like secondlife:///app/teleport/Endora/123/234/345

Message parameters

```
msg:url     = This link brings you at target location:
msg:lm      = This landmark brings you at target location...
msg:return  = This link brings you back:
msg:norelay = Please, use this item in order to use me...
```

The params '**msg:...**' take text values which introduce given files and SLURLs. They also accept placeholders, section 1.3.2.

The param '**msg:url**' saves the text to introduce the target SLURL. The message for traveller is a combination of this text with the SLURL, sent in a single IM.

The param '**msg:lm**' saves the text to be sent if a target LM is given. The message and the landmark are sent separately.

The param '**msg:return**' saves the text to introduce the return SLURL.

The param '**msg:norelay**' saves the text sent if the teleporter detects missing of the relay and sends file noted under '**norelay**' param.

1.3.3 Target sections

A target section provides three Entries: The *target name*, the *target image* and the *target definition*. Since TPSign has no possibility to select targets, only the first target is selected permanently, even if more than one target section is given. Hence, only a single section makes sense for TPSign, but you can define up to 80.

Target name

```
[Welcome to Endora]
```

The target name is the name of the section. It must start by a letter or underscore and can be off multiple words. It must be distinct and 32 letters at last. The target name can be injected via placeholder '**%n**' into the title (hover text) and sent messages.

Target image

```
image       = 8cee492c-917e-341d-b3a9-63d83ed44d34
```

The param is used as target picture. The param '**image**' is optional, if omitted, the image defined in main section is used instead. The value can be an UUID or an image in object inventory, but than it must be fullperm.

Target definition

```
target      = Endora/127/169/42
```

The param '**target**' is required to localise the target, the value can be a LM or a SLURL.

If this is a LM, it muss be in in the object inventory. In this case the LM is given to the user if LM mode is active, section 1.3.2. This way they can visit the place later by using the landmark.

If the target is defined by a SLURL, user's inventory is not overfilled by sent LMs, but the traveller can visit the place again only via the teleport history or manually taken LMs. The SLURL can be given by three manners:

- SLURL: http://slurl.com/secondlife/Endora/127/169/42
- MapURL: http://maps.secondlife.com/secondlife/Endora/127/169/42
- RawURL: Endora/127/169/42

RawURL starts with the sim name and thus it is universal. **Note**: The RTS documentation may call this form 'Stripped SLURL', the actual name was met later.

1.4 Extension

TPSign is a RTS teleporter, but it uses not all possibilities RTS offers: TPSign was originally developed as a teleporter for a single destination. RTS allows to define multiple destinations, but for traditional reasons, TPSign offers no possibility to select one of them.

In this section we extend the teleporter by the missing ability. We don't have to write or change scripts. All we'll do, is just linking properly named prims to the teleporter. The result (but not the goal) of the extension is a teleporter called TPVendor, actually available in the RTS package, figure 1.5.

Figure 1.5: TPVendor

This section is written for purchaser of TPSign – who can travel grid-wide, very soon needs a teleport device for many destination around the grid. But also for purchaser of the RTS package, as a demonstration of RTS and tutorial for working with this framework.

1.4.1 TPSign architecture

First, we take a brief look at the design and working way of TPSign. This may help to understand how RTS teleporters work. Detailed information is given in the RTS documentation, here we really take an overview.

Files

This files are instaled into TPSign:

Script '**.core**': The core script, controls other scripts, performs teleports and enforces sending of messages and files. It also opens the owner menu and runs a number of core services.

Script '**.config**': Configuration script. Reads the '**config**' notecard and shares the read data over other scripts. Other script have no access to this notecard, which simplifies the script code.

Script '**.sender**': Sender script. Sends files like LMs or the teleport relay and messages vie IM as well, to the user and device owner, like error messages produced while configuration and run time.

Script '**.targets**': Target list. Can take up to 80 targets given in the configuration. Manages the target data: Stores the specified data, resolves the global position, provides data for a selected target.

Object '**TP Relay (r upper arm)**': The teleport relay, as given in the configuration, param '**norelay**' (section 2.3.2.)

Core services

RTS separates the functionality of the teleporter in single services. The services are executed by the core script, but are transferable. TPSign uses this services:

The *text service* is responsible for displaying the title (hover text.)

The *message service* sends error messages to the owner via IM, the messages are relied to the sender script, which resends them to the owner.

The *image service* displays the target picture.

The *touch service* starts the teleport if the teleporter is touched. This saves up a prim

The *menu service* opens the owner menu by the longclick.

1.4.2 Design

Back to the extension challenge. The teleporter is a linkset while the prims have roughly four functions. The *base prim* runs all scripts and is in fact the unchanged TPSign.

The *thumbnail prims* and the *scroll buttons* are used to select destinations. RTS offers also other ways to select the teleport destination but we use these two.

The *menu button* makes opening the owner menu much easier than the longclick. Finally added background prim keeps together all others visually, but is not important for the teleporter itself.

Base prim

At first, we need a base teleporter: We rezz TPSign and set the prim size to (0.01, 1.52, 1.52), figure 1.6. The 2cm enlargement go up to the distance between later following thumbnail prims. The prim we rename now – to '*TPVendor*'.

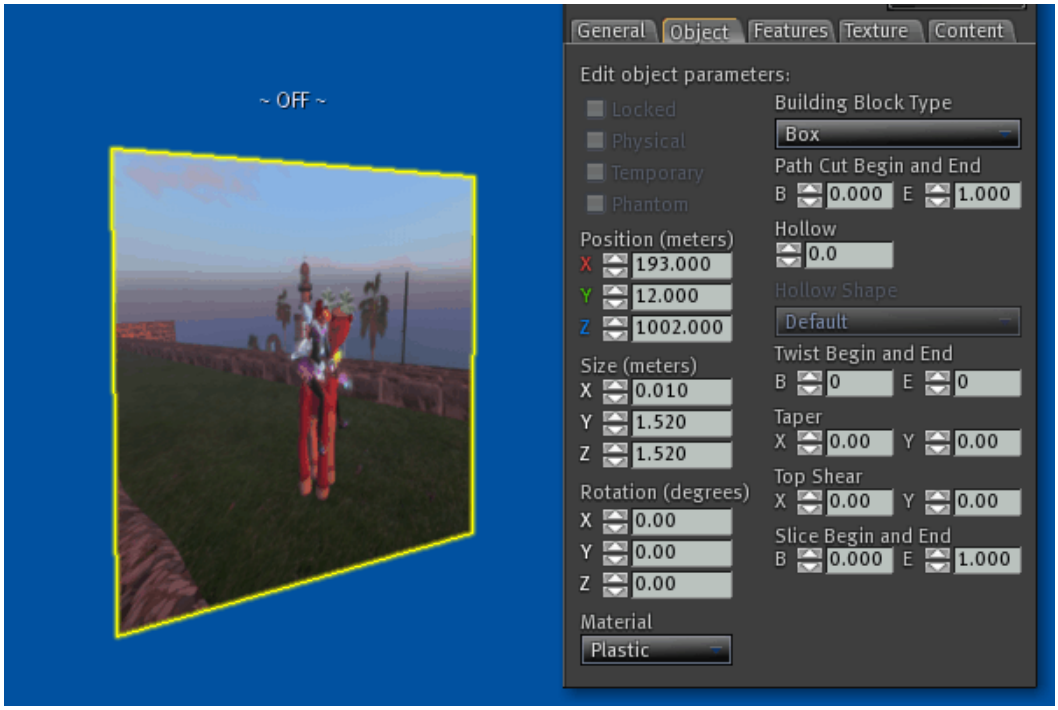

Figure 1.6: Rezzing the base

Alternatively, we could also create a prim box, in the right size and install required files into it, section 1.4.1.

Thumbnail prims

We use 6 of them (RTS supports 20.) The prims display target pictures, a click at one of them selects the shown destination by displaying it on the base prim.

We create a prim box in the size (0.01, 0.5, 0.5) and align it near the base: We copy the x and z values from the base position, the y value we copy and increase by 1.20 meters. The prim box we copy now towards top: Edit, press and hold the shift key and pull the prim on the blue arrow, at the place where it was a copy of the prim box appears. Repeat this by copying the prim towards bottom.

Both copies we align 51cm above and below the original. All three prims we copy at same wise towards side and align on the other side of the base prim.

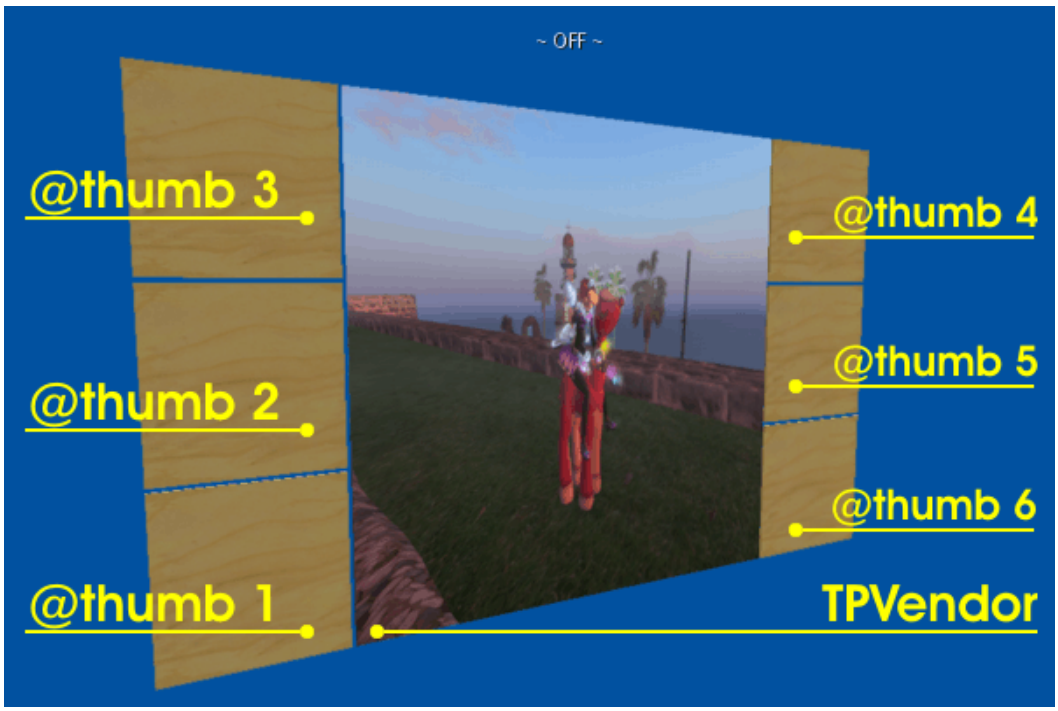

Figure 1.7: Thumbnail prims

In order the prims are identified to be thumbnails, we must name them properly: The prim name starts by '**@thumb**', followed by a number. The number defines the order in which the prims take the preview pictures, figure 1.7.

Scroll buttons

Scroll buttons are two prims, which move the target selection by a certain number of targets. A prism looks nearly as an arrow and not needs a texture.

We create a prism of the size (0.12, 0.12, 0.02) and justify it below the left thumbnail. The prism we copy at right and justify below the right thumbnail, figure 1.8.

In order the scripts identify the prims as scroll buttons, we have to name them properly, too. To scroll forward, the prim name must start by '**next**', following by a number of positions to scroll. To scroll backward, the prim name must start by '**prev**', following by a number of positions.

We have 6 thumbnails and a target prim, thus 7 target pictures are shown ar same time. Hence, the scroll buttons must scroll by 7 positions – we name them '**prev7**' and '**next7**'. Both prims will get later also a blank texture and blue color.

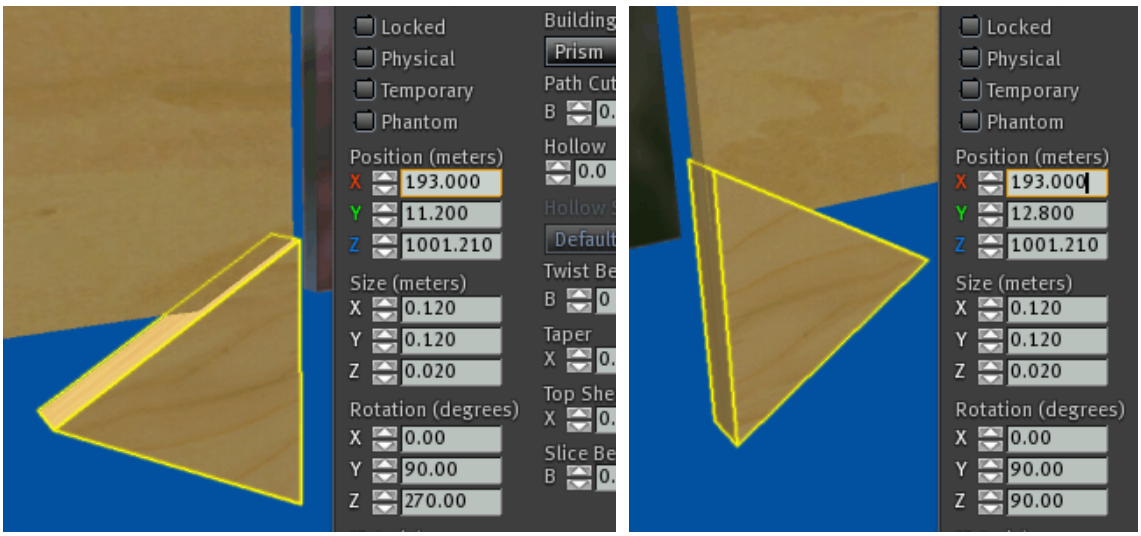

Figure 1.8: Scroll buttons

Menu button

The menu button is a prim which takes our logo, and if clicked, opens the owner menu. This allows to avoid the less convenient longclick. We can create a new prim, but the simplest way is to copy the right scroll button towards left, justify it below the base prim, change the form to a box and give proper size, figure 1.9.

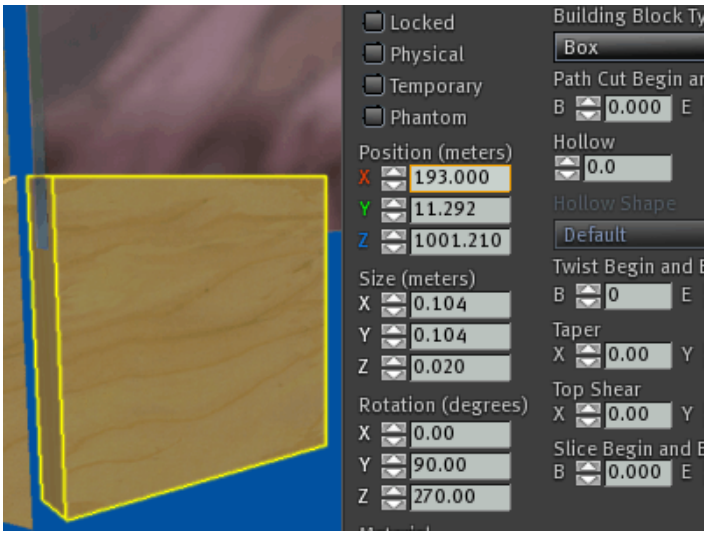

Figure 1.9: Menu button

You guessed right: The prim must have a proper name in order it works as a menu button. In this case, the name is exactly given: '`.menu`' (the dot is no typo.)

Finalisation

We have almost finished. We miss just a background prim. Here we mist not care about anything except aligning it properly behind other prims. Finally, we link all prims together in a way the base prim becomes root. In one step we can now polish and brighten the teleporter. The result is shown in the figure 1.10.

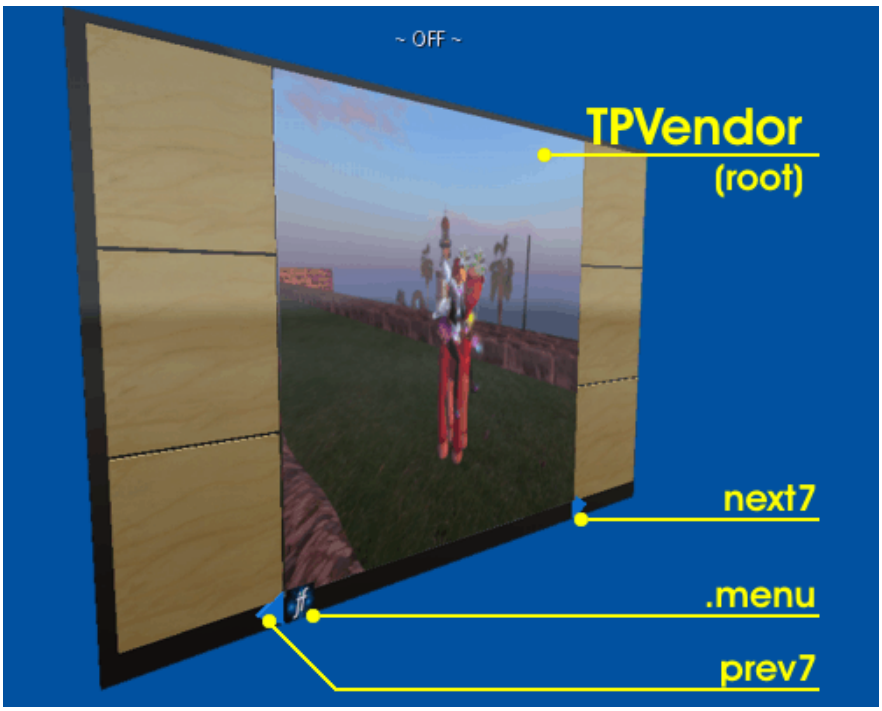

Figure 1.10: Finalization

1.4.3 Configuration

The design is complete. Last step is extending the configuration. To do so, we edit the teleporter and the '**config**' notecard inside.

Here we simply change the title (to reflect multiple targets) and replace a single target section by a list of sections – each one for every teleport destination. We can enter here up to 80 of them. The TPVendor is already configured by 11 example destinations, figure 1.11.

Please, compare it with the TPSign configuration, figure 1.4. As we can see, in the main section we changed only the title, and there are more target sections. Target names receive the rating (G, M, A) of the sim. The target name is inserted via the placeholder '**%n**' into the title, this way if we use the teleporter, we can see the sim rating before we go to the location.

```
# TPVendor Configuration
[*MAIN]
title       = Destination %p of %c\n%n\n...touch to visit
# --- The rest like for TPSign -------------------------

# Target list
[Linden Playground (M)]
image       = 1b5e8133-3fb8-5488-c498-e63ec41b2010 # Da Boom (143, 148, 41)
target      = Da Boom/128/128/35

[Protected Ocean (G)]
image       = 16c62e5b-0b9c-9d78-1aba-e1f45af68bdb # Pravatch (230, 229, 2)
target      = Pravatch/221/227/3

[MYST Island (G)]
image       = 5a6b672a-6c6a-d2a1-a020-04217cdaf0da # Age of Myst (171, 46, 33)
target      = Age of Myst/175/14/24

[Great Wall Of China (G)]
image       = 10594b0c-2ae4-3f7b-052b-a77a098fefcd # China Sichuan (130, 10, 24)
target      = China Sichuan/133/16/24

[Museum of Illusions & Magic (G)]
image       = 76999bdb-11ba-c08d-4d5a-24ea1bed073a # AmazingIllusionMuseumEI
target      = Enchantment Island/59/120/45

#This target was in TPSign configuration
[Welcome to Endora (A)]
image       = 8cee492c-917e-341d-b3a9-63d83ed44d34
target      = Endora/127/169/42

[The Grand Canyon (M)]
image       = d835568f-fb32-5fe4-2ba1-99191261eced # Grand Canyon (106, 173, 110)
target      = Grand Canyon/89/186/110

[Happy Mood (G)]
image       = 525ea54c-49dd-bc1d-327e-c64cd90610db # HappyMood (107, 144, 79)
target      = HappyMood/109/144/80

# --- 3 more target sections follow --------------------
```

Figure 1.11: Configuration of TPVendor

Target pictures are given by the UUID. The comment char allows to put the picture name here. The picture name is not meant for the teleporter, it allows to find the picture in our inventory later.

The target place is given by RawURL. This way we must not take a LM to prepare the destinaton, but can use the teleport history or simply take the sim name and position from the taken screen shot.

After the configuration text is extended and saved, we have to reset the teleporter: Click on the menu button, than '**reset**' button in the opened menu. After the configuration is complete we can start the teleporter via the owner menu again. The device is ready to use, as shown in figure 1.5.

1.4.4 Extended architekture

We have no script changed, added or removed in the original teleporter, so why it works as it works? The answer is in the architecture and interfaces of RTS. This section should explain how they work together in the context of TPVendor. For more information, please use the RTS documentation.

The targetlist script '**.targets**' manages the destinations given in the configuration. If one is selected on any way, the script provides a set of data for the target, including the target picture and pictures before and after the selected in the list.

By adding and naming of prims, we activated two more core services, the *thumb service* and the *button service*.

The name prefix '**@thumb**' denotes prims as thumbnails. Than they fall under control of the *thumb service*. If a target is selected, this service displays the pictures of the neighbour-targets on the thumbnail prims. If one of them is clicked, the service produces a control message, which moves the target selection in a way the target is selected that was shown on the clicked prim.

The prims with name '**prev7**', '**next7**' and '**.menu**' are meant to be buttons and are covered by the *button service*. The service produces a control message, if one of the prims is clicked.

If the button name starts by a dot, the produced message is meant for the core script. The menu button has the name '**.menu**', this produces a control message opening the owner menu.

If a button name starts by a letter, the control message is meant for the targetlist script. The name prefix '**prev**' and '**next**' enforce the script to move the target selection by the number after the prefix. The scroll buttons enforce the selection to move by 7 targets this way. The thumb service uses same control messages and enforce the targetlist to move the selection by 1, 2 or 3 targets.

Hence, in this design, TPVendor uses only one possibility to select the destination. But the targetlist script offers a few more: The exact target number to select (button prefix '**pos**'), selection of the first or last destination (buttons '**first**', '**last**'), Random target selection (button '**random**') and a free text search.

That's all. I wish you much fun and success with experimenting with the teleporter and RTS.

2 Deutsch

2.1 Einleitung

TPSign ist ein Teleportgerät, das mittels der RTV Technologie gridweit funktioniert. TPSign war bereits als RLV Teleporter entwickelt, jetzt, in der Version 2, wurde er mit der RTS Technologie neu aufgebaut. RTS steht für '*RLV Teleporter System*', es ist ein Framework zum Bau von RLV Teleportern. TPSign erbt auf diese Weise viele Features von RTS.

Was ist an TPSign anders? TPSign ist ein gridweiter Teleporter. Er erreicht jeden Ort, der via einer LM erreichbar ist. Der Avatar wird nicht physikalisch bewegt, der Teleport bricht nicht ab, wenn die Zielposition auf der anderen Sim oder Kontinent liegt. TPSign öffnet auch keine Weltkarte, wie Kartenteleporter das tun. Statdessen überreicht TPSign die Zielkoordinate an den Viewer und löst den unmittelbaren Teleport aus.

Grundsätzlich, benötigen RLV Teleporter, dass der Viewer eine spezielle RLV Schnittstelle unterstützen, um die Zielkoordinate zu übergeben. TPSign funktioniert auch auf diese Weise. Da aber nicht jeder Viewer diese Schnittstelle anbietet, funktioniert TPSign auch ohne sie: Man kann zum Einem den RLV Support ausschalten und den Teleporter dadurch zu einem Landmarkverteiler machen.

Zum Anderen macht das mitgelieferte TP Relay jeden RLV Teleporter (und auch den TPSign mit aktiven RLV Support) zu einem Kartenteleporter. Dies erlaubt es, RLV Teleporter, TPSign und auch Kartenteleporter auf dieselbe Weise und mit jedem Viewer zu benutzen. Aus diesem Grund überreicht TPSign dieses Relay an den Benutzer, sofern er kein RLV oder kompatible Viewer einsetzt.

2.1.1 Zielgruppe

Für wen ist TPSign geeignet? Nun, er erlaubt zum Ort auf derselben Sim zu springen, ob eine Skybox, Club oder die nächste Etage im Haus. Das sind aber lokale Teleports und es gibt einfachere und leichtere Lösungen für diese Aufgabe.

Möchte man aber jeden Ort auf der Grid mit nur einem Klick erreichen, ob man ein Gerät für Lieblingsrte braucht, oder ein Netzwerk von Partnerteleporter, egal ob die

Sims verbunden sind oder über die Grid verteilt, dann ist TPSign eines der Geräte der Wahl.

RTS wurde entwickelt, um mehre Teleportziele zu verwalten. Aus traditionällen Gründen unterstützt TPSign aber nur ein Teleportziel. Mit ein wenig Aufwand kann man ihn aber um die Mehrzielfähigkeit erweitern. Im Paket ist ein schrittweiser Tutorial enthalten wo das behandelt wird, so wie das Ergebnis: TPVendor ist ein RTS Teleporter, für bis zu 80 Teleportziele.

2.1.2 Struktur

Die Dokumentstruktur ist die folgende: Die *Einleitung* stellt den Inhalt des Produktpakets vor und wie Sie das Produktupdate erhalten. Es wird erklärt, was RLV ist, warum diese Schnittstelle benötigt wird, welche Viewer sie unterstützen und wie man TPSign ohne dieser Viewer nutzen kann.

Der Abschnitt 2.2 behandelt die *Installation* des Teleporters und seinen Einsatz. Seine *Konfiguration* wird im Abschnitt 2.3 erklärt.

Anschließend folgt der Abschnitt 2.4 mit einer kurzen Einführung in die Arbeitsweise des Teleporters, und einer schrittweisen Anleitung zum Erweitern von TPSign zu TPVendor, einem Teleporter, der mit einer Liste von Zielen arbeitet, und somit die Mehrzielfähigkeit von RTS ausnutzt.

TPSign ist ein RTS Teleporter. Deshalb ist die RTS Dokumentation für Hintergrundinformation wichtig, die über die beiden letzten Abschnitte hinaus geht. Auch in diesen Abschnitten wird darauf verwiesen. Die Dokumentation steht als eine PDF Datei zur Verfügung, verlinkt auf der RTS Produktseite.[1]

2.1.3 Paketinhalt

Inhalt des Produktpakets ist in der Abbildung 2.1 dargestellt. Enthalten sind unter Anderem der Teleporter selbst, so wie seine Erweiterung, *TPVendor*. Beide sind einsatzfertig, benötigen aber eine Rekonfiguration, etwa zur Einstellung von Teleportziele.

Das Benutzerhandbuchist in der Notekarte *TPSign User Manual* zu finden, und die erforderliche RTS Dokumentation in der Notekarte *RTS User Manual*. Beide enthalten weitere Notekarten mit Teildokumentation und in der deutschen Sprache.

[1] https://marketplace.secondlife.com/p/RLV-Teleporter-System-private-edition/2944310

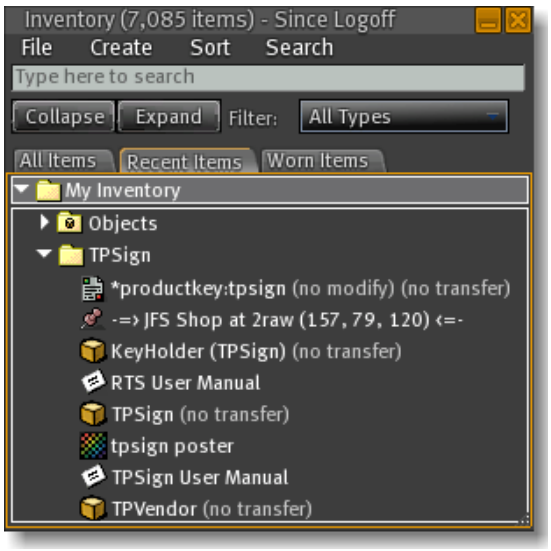

Figure 2.1: TPSign, Paketinhalt

Auspacken

Auspacken können Sie das Paket auf traditionelle Weise und automatisch: Im ersten Fall rezzen Sie es am Boden, öffnen und kopieren Sie den Paketinhalt. Sie erhalten dann ein Verzeichnis namens '**TPSign (BOXED)**'.

Alternativ können Sie das Paket sich selbst öffnen lassen. Dazu rezzen Sie es am Boden oder tragen in der Hand (falls rezzen nicht erlaubt ist.) Sind Sie auf einem Ort mit erlaubter Skriptausführung, öffnet sich das Paketmenü. Ignorieren Sie es, können Sie das Paket anklicken und erhalten Sie das Menü erneut.

In diesem Menü brauchen Sie den Button '**Open**'. Nach dem Klick darauf übergibt die Box das Produktverzeichnis '**TPSign**'. Anschließend öffnet sich das Paketmenü erneut, wo Sie den '**Remove**' Button anklicken können. Er zerstört die Box, falls sie gerezzt ist oder legt sie ab, wenn getragen.

Bitte bewahren Sie die Produktbox als Sicherheitskopie auf, oder wenigstens den Skript '***productkey**'. Er ist zum Anfordern von Produktupdates erforderlich. Es steht auch ein Onlinetutorial zum automatischen Entpacken von JFS Produktboxen zur Verfügung.[2]

Update

TPSign befindet sich derzeit in einem Verkaufssystem, das eine automatische Auslieferung von Updates erlaubt. Möchten Sie dagegen Updates manuell anfordern,

[2] http://jennassl.blogspot.com/2011/11/unpacking-jfs-boxes.html

haben Sie zwei Möglichkeiten: Via der Produktbox selbst oder des mitgelieferten KeyHolders.

In beiden Fällen wird der '***productkey**' Skript aktiv, der Updates vom Updateorb anfordert. Der Orb muss dabei in der Nähe sein, deshalb müssen Sie dafür erst einen der JFS Shops aufsuchen, wo der UpdateOrb installiert ist. Die Liste solcher Shops finden Sie in einem dafür eingerichteten Blogartikel.[3]

Anstatt der *Produktbox* können Sie auch jedes Prim nehmen, in das der Skript '***productkey**' installiert ist. So fordern Sie Produktupdates damit an:

Kommen Sie näher als 10m an den Updateorb. Rezzen Sie nun die Produktbox, das Menü erscheint. Wenn nicht, bitte die Box anklicken. In diesem Menü klicken Sie bitte den '**Update**' Button. Der Updateorb wird das Update ausliefern. Im darauf geöffneten Menü klicken Sie bitte den '**Remove**' Button, damit zerstört sich die gerezzte Box automatisch.

Der *KeyHolder* ist ein armbandähnliches Gerät, das Schlüssel für bis zu 12 Produkte verwaltet und das Herumschleppen von Produktboxen überflüssig macht. Der Skript '***productkey**' muss ins Gerät installiert sein, damit er arbeitet.

Im mitgelieferten KeyHolder ist der Produktschlüssel bereits installiert. Um damit die Produktupdates anzufordern, verfahren Sie bitte wie folgt:

Kommen Sie ebenfalls in die Nähe des Updateorbs. Tragen Sie den Keyholder und klicken den an, ein Menü erscheint. Bitte wählen Sie das RTS Produkt aus, falls nicht bereits ausgewählt. Nach einem Klick auf den '**Update**' Button liefert der UpdateOrb das Update aus. Sie können den KeyHolder wieder ablegen.

In beiden Fällen ist das Update eine Nachlieferung des aktuellen Produktpakets. Es steht ebenfalls ein Onlinetutorial zum Updaten von JFS Produkten zur Verfügung.[4]

2.1.4 Warum RLV Teleporter

RLV war ursprünglich ein spezieller Viewer, der mit dem Ziel entwickelt wurde, Immersion (das Gefühl, im Spiel zu sein) des Benutzers zu unterstützen. Hierfür bietet der Viewer eine besondere Schnittstelle, die den Skripten erlaubt, den Viewer eingeschränkt zu kontrollieren.

Mithilfe dieser Schnittstelle können Skripte einige Aktionen des Viewers auslösen, die sonst nur der Viewerbenutzer ausführen kann, wie etwa vom gesessenen Objekt aufstehen oder zu einem Ort teleportieren. Die Liste von ausführbaren Aktionen ist

[3] http://jennassl.blogspot.com/2010/05/actual-shop-list.html
[4] http://jennassl.blogspot.com/2011/11/updating-jfs-products.html

relativ umfangreich, wobei TPSign von allen Features von RLV nur den Teleportbefehl umsetzt.

Erreicht dieser Befehl den Viewer, setzt er sich sofort in Bewegung. Der Benutzer sieht nur den Ladebalken und kurze Zeit später den Avatar am Zielort stehen. Genau so, wie beim Benutzen einer Landmarke oder Weltkarte, nur ganz ohne eines geöffneten Landmarkenfenster oder Kartenfenster.

Genau darin liegt der Vorteil von RTS Teleportern (oder allgemein der RLV Teleportern) gegenüber Landmarkverteilern oder Kartenteleportern: Der Benutzer wird aus der Spielsituation durch das geöffnete Landmark- oder Kartenfenster nicht herausgerissen.

Ursprünglich wurde die RLV Schnittstelle nur durch den RLV implementiert, jedoch seit die Entwicklerin des Viewers sie veröffentlicht hat[5], konnten andere Hersteller ihre Objekte RLV-fähig machen und so die Verbreitung des RLV fördern. Inzwischen steht diese Schnittstelle für alle Viewerentwickler als *RLVa* zur Verfügung[6] und wird von vielen Drittanbieterviewern implementiert.

Ohne RLV

Ein Nachteil von RLV Teleportern ist, dass nicht jeder Viewer diese Schnittstelle unterstützt. Benutzer eines Viewers ohne RLV Schnittstelle kann solche Teleporter nicht auf die beschriebene Weise nutzen. Vor allem der offizielle Viewer kennt die RLV Schnittstelle nicht.

Beim Entwickeln von RTS wurde dies berücksichtigt: RTS Teleporter, daher auch TPSign, lassen den RLV Support in der Konfiguration ausschalten. Der Teleporter funktioniert dann mit jedem Viewer, spricht jedoch das RLV Interface nicht an, sondern übergibt lediglich die LM oder SLURL zum Zielort.

Mit einem Teleportrelay

Eine andere Möglichkeit, RLV Teleporter (daher RTS Teleporter und daher TPSign) mit einem Viewer zu benutzen, der die RLV Schnittstelle nicht hat, bietet das Teleportrelay.

Zum Nutzen von RLV Teleporter ist generel neben dem RLV-fähigen Viewer, auch ein spezielles Objekt erforderlich, das Relay. RLV besitzt eine technische Sicherheitschranke: Es werden nur RLV Befehle angenommen, die von Objekten kommen, die dem Viewerbenutzer gehören. RLV Teleporter sind Inworldobjekte, die zumeist anderen gehören und so den Viewer gar nicht direkt ansprechen können.

[5] http://wiki.secondlife.com/wiki/LSL_Protocol/RestrainedLifeAPI
[6] http://rlva.catznip.com/blog/2009/10/release-rlva-1-0-5/

Das Relay ist ein Objekt, das dem Viewerbenutzer gehört und am Avatar oder HUD getragen wird. Es empfängt RLV Befehle von RLV Geräten und leitet die Befehle zum Ausführen an den Viewer weiter. Meistens führt das Relay dabei auch eine Sicherheitprüfung durch: Darf das Gerät den Viewer kontrollieren oder nicht? Im Zweifelsfall fragt das Relay den Benutzer via Menü, steht das Gerät oder Gerätebesitzer auf einer schwarzen (oder weißen) Liste, verweigert oder akzeptiert das Relay automatisch.

Übliche Relays folgen der Spezifikation, sie setzen die Benutzung eines RLV-fähigen Viewers voraus und leiten alle RLV Befehle der Inworldobjekte an den Viewer weiter. Um RLV Teleporter nutzen zu können, sind aber nur drei Befehle von Bedeutung: Die *Versionsprüfung*, der *Unsitbefehl* und der eigentliche *Teleportbeffehl*.

Nur diese Befehle werden vom Teleportrelay an den Viewer weitergeleitet, alle anderen Befehle werden abgewiesen. Dadurch kann man den RLV ohne Gefahr einer unerwünschten Aktion seitens RLV Geräte nutzen. Bei anderen Relays ist dafür eine Auseinandersetzung mit deren Sicherheitseinstellungen erforderlich.

Wird dabei ein Viewer benutzt, der die RLV Schnittstelle nicht hat, emuliert das Teleportrelay die schnittstelle, indem es eine gestellte Versionsantwort an das Teleporter versendet und zum Teleportieren das Kartenfenster öffnet.

Auf diese Weise bekommen Inworldobjkte eine stark eingeschränkte Kontrolle über den RLV-fähigen Viewer. Mit Viewern, die RLV nicht verstehen kann man RLV und RTS Teleporter trotzdem nutzen – sie werden zu normalen Kartenteleporter. Das Relay wurde mit diesem Ziel geschaffen und ist im TPSign installiert, zum Aushändigen, wenn der Benutzer kein Relay benutzt.

RLV-fähige Viewer

Es gibt mehrere Viewer, die RLV Schnittstelle implementieren und RLV Teleporter mit einem üblichen Relay nutzen lassen. Einige davon sind aufgelistet:

- Klasisches RLV (http://www.erestraint.com/realrestraint/)
- Cool Viewer (http://sldev.free.fr/)
- Firestorm und Phoenix Viewers (http://www.phoenixviewer.com/)
- Imprudence (http://blog.kokuaviewer.org/)
- Rainbow (http://my.opera.com/boylane/blog/)
- Singularity (http://www.singularityviewer.org/)

Die Liste dient dem Überblick und wird nicht vollständig sein.

2.1.5 Globale Koordinaten

Jeden Ort in SL kann man mit einer lokalen und einer globalen Koordinate angeben. Lokale Koordinate beschreibt die Position eines Punktes bezüglich eines speziell festgelegten Ursprungs.

Oben in der Symbolleiste kann man etwa die lokale Position des Avatars bezüglich Ursprung der Sim einsehen. Sei es ein Punkt mit der Koordinate **<128, 128, 24>** (genau in der Mitte der Sim.) Auf der ganzen SL-Welt gibt es so viele Punkte mir der lokalen Koordinate **<128, 128, 24>**, wieviele Sims es gibt.

Erst mit Angabe der Sim kann eine Lokalkoordinate den Punkt eindeutig positionieren. Eine SLURL gibt deshalb die lokale Position, so wie die Sim an, von der aus die Position gemessen wurde. Solch eine beschreibung ist in der SL Welt eindeutig:

http://slurl.com/secondlife/DaBoom/128/128/24.

Alle Sims fügen sich zu einer *Grid* zusammen, das ist die Welt von SL. Jede Sim besitzt deshalb eine globale, d.h. für die Grid eindeutig bestimmte Koordinate ihres Ursprungs. Es lässt sich etwa ermitteln, dass der Ursprung der Sim *Da Boom* die globale Position **<256000, 256000, 0>** hat. Wie kommt man auf Globalkoordinaten einer Sim? Z.B. über den Dienst *Grid Survey*, mit diesem Aufruf:

http://api.gridsurvey.com/simquery.php?region=DaBoom

Die Antwort enthält zwei Angaben: **x=1000**, **y=1000**. Diese Werte multiplizieren wir mit 256 (die Größe einer Sim), und nehmen **z=0** an. Fertig.

Kombiniert man nun die lokale Koordinate eines Ortes auf der Sim mit der globalen Koordinate ihres Ursprungs, erhält man die globale Koordinate des Ortes. So befindet sich der Punkt '**Da Boom/128/128/24**' auf einer Position **<256128, 256128, 24>**, gemessen vom Ursprung der Grid, d.h. global.

Die globale Position ist eindeutig in der ganzen Grid, sie ist außerdem einfach (nur eine Vektorzahl), aus diesen Gründen arbeiten Kartenteleporter und RLV/RTS Teleporter mit dieser Angabe.

Die globale Position hat aber einen Nachteil: Gelegentlich kommt es vor, dass die Sim sich verschiebt. Dann erhält sie eine andere Ursprungskoordinate. Dadurch ändern sich die globale Koordinaten aller Orte auf der Sim. Teleporter, können die Orte nach der Verschiebung via alter Koordinaten nicht mehr erreichen, die Zielposition muss erneut berechnet werden.

Einige Teleporter berechnen deshalb die Zielposition für jeden Teleport neu. RTS (daher auch TPSign) verzichtet auf die permanente Neuberechnung (sie belastet die Sim) sondern cacht den Wert und erneuert ihn erst wenn die Berechnung zu lange zurück liegt. Außerdem kann der Gerätebesitzer den gecachten Wert als veraltet kennzeichnen und so beim nächsten Teleport berechnen lassen.

2.2 Installation und Gebrauch

Figure 2.2: Installation

TPSign ist ein Gerät, das nur ein Prim belegt, welches als eine quadratische Fläche geformt ist. Die Installation erfolgt in drei Schritten:

1. Rezzen Sie den TPSign und positionieren es am Einsatzort, etwa an einer Wand, auf einem Tisch oder Pult, Abbildung 2.2. Beim Besitzerwechsel initialisiert sich das Gerät automatisch. Dieser Vorgang dauert wenige Sekunden.

2. Optional können Sie den Teleporter konfigurieren und das Teleportziel festlegen, Abschnitt 2.3.3.

3. Nun aktivieren Sie das Gerät, wie im Abschnitt 2.2.1 beschrieben.

Der Teleporter muss nicht seine rechteckige Form behalten. Sie können es genauso gut zu einer Kugel oder zu einem runden Knopf umformen oder ihm gar eine gesculptete Form geben. In allen Fällen bleibt es ein Teleportergerät, das aufs Anklicken reagiert und die anklickende Person zum eingestellten Ziel befördert. Im Abschnitt 2.4 wird erklärt, wie man den Teleporter auch so erweitert, dass man vor dem Teleport auch das Ziel auswählen kann.

2.2.1 Inbetriebnahme

Der TPSign ist mit einem Teleportziel vorkonfiguriert und kann sofort nach dem Auspacken eingesetzt werden, allerdings teleportiert er zum voreingestellten Ziel. Falls Sie die RTS Teleporter noch nicht kennen, empfiehlt es sich, den TPSign in der vorkonfigurierten Form auszutesten und erst dann die Konfiguration zu ändern.

Bedient wird der Tleporter über das Besitzermenü, das nur dem Besitzer angezeigt wird. TPSign besteht aus nur einem Prim (kein extra Prim als Menübutton), daher lässt sich das Besitzermenü nur via Langklick öffnen.[7] Das Menü enthält vier Buttons:

- '**start**', '**stop**' um den Teleporter starten oder anzuhalten.
- '**reset**': Initialisiert alle Skripte. Hierdurch wird die Konfiguration eingelesen und das Gerät eingestellt. Diesen Vorgang müssen Sie deshalb nach Änderungen in der Konfiguration oder am Teleporter erneut auslösen.
- '**refresh**': Dieses Button löscht die gespeicherten globale Zielkoordinate (Abschnitt 2.1.5) und erzwigt so ihr Neuberechnen beim nächsten Teleport.

2.2.2 Gebrauch

Um mit TPSign zu reisen, reicht ein Klick darauf. Was dann passiert, hängt davon ab, ob der Teleporter in einem RLV oder LM Modus eingestellt ist (ausgeliefert ist er mit dem aktiven RLV Modus) und ob der Benutzer einen RLV-fähigen Viewer und das Relay einsetzt. Der genaue Ablauf ist in der Abbildung 2.3 dargestellt.

Im RLV Modus

Ist der RLV Modus aktiv, versendet der Teleporter eine Teleportanfrage mit Angabe des Benutzers, die Anfrage empfängt sein getragenes Relay.

Ist die Anfrage akzeptiert, leitet sie das Relay an den Viewer weiter. *Der Viewer führt den Teleport aus.* Der Teleporter empfängt dann die positive Rückmeldung des Relays und (falls eingestellt) übergibt die SLURL eigener Position an den Benutzer per IM (die Rückkehr-SLURL.)

Weist das Relay die Anfrage ab, kann der Teleporter nur noch die LM oder SLURL zum Ziel an den Benutzer übergeben, falls der LM Modus aktiv ist.

Ist das Relay dagegen nicht vorhanden, ausgeschaltet, oder hat der Benutzer seine Rückfrage nicht rechtzeitig beantwortet, stellt der Teleporter das Timeout fest und händigt das Teleportrelay (oder eine andere in der Konfiguration angegebene Datei aus) dem Benutzer aus.

[7] Bitte den Teleporter anklicken und die Maustaste erst nach einer Sekunde los lassen.

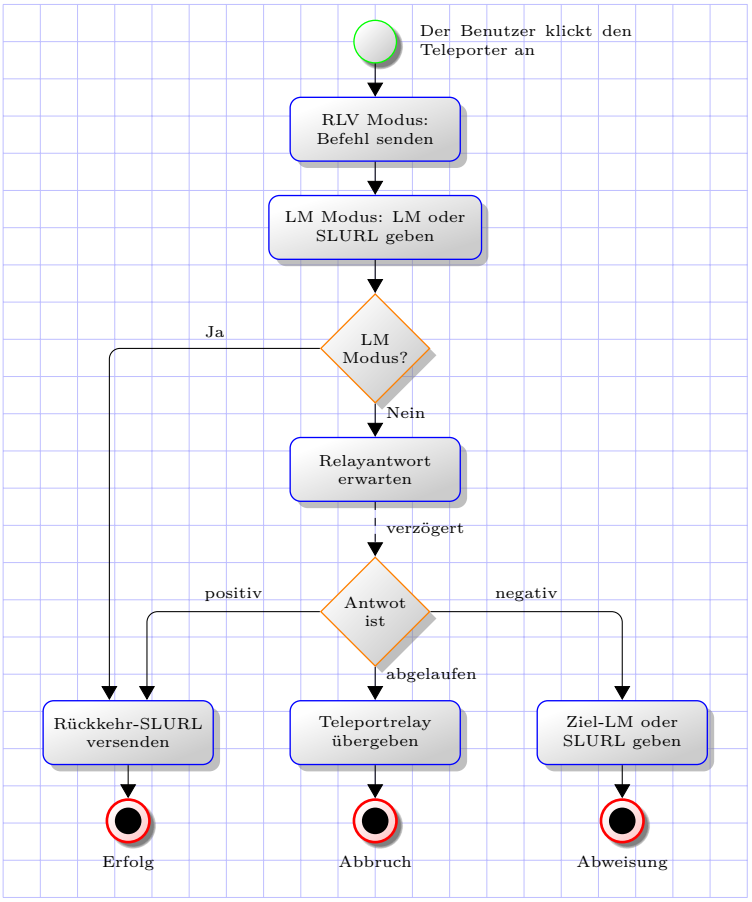

Figure 2.3: Teleportvorgang

Im LM Modus

Ist das LM Modus aktiv (unabhängig vom RLV Modus), wird die LM oder SLURL zum Ziel an den Benutzer übergeben. Der Teleport gilt dann als erfolgreich, da der Benutzer die LM bzw. SLURL benutzen kann.

Ist gleichzeitig der RLV Modus aktiv, wird die Teleportanfrage dennoch versendet und evtl. durch den Viewer ausgeführt, auf die Rückmeldung des Relays wartet der Teleporter dann nicht mehr.

2.3 Konfiguration

Die Konfiguration von TPSign wird (wie bei RTS Teleportern) in einer Notekarte vorgenommen, die im Objektinventar des Teleporters unter dem Namen '**config**' vorliegt. Beim Ausliefern ist der TPSign so konfiguriert, wie in 2.4 abgebildet.

```
# TPSign Konfiguration
[*MAIN]
user         = all

image        = 09BB628B-F97A-44FD-9503-6E4EC62DE59E
side         = all

title        = %n\n...touch to visit
color        = lavender
fadeout      = 0

mode         = RLV
fail         = LM
return       = LM
delay        = 10

norelay      = TP Relay (r upper arm)
offset       = -3
fasttp       = OFF
refresh      = 24
format       = APP

msg:url      = This link brings you at target location:
msg:lm       = This landmark brings you at target location...
msg:return   = This link brings you back:
msg:norelay = Please, use this item in order to use me...

# Das einzige Ziel
[Welcome to Endora]
image        = 8cee492c-917e-341d-b3a9-63d83ed44d34
target       = Endora/127/169/42
```

Figure 2.4: Konfiguration von TPSign

Für die ausführliche Erklärung der Gerätekonfiguration im Allgemeinen steht die RTS Dokumentation zur Verfügung. Hier wird lediglich auf den Konfigurationstext eingegangen und somit auf Einstellung speziell des TPSign.

2.3.1 Syntax

Das Zeichen '**#**' leitet das Kommentar ein: Das Zeichen selbst und der nachfolgende Text bis zum Zeilenende wird ignoriert. Der Text vor dem Kommentarzeichen wird beachtet (außer Leerzeichen), Kommentare werden immer mit dem Zeilenende abgeschlossen.

Leere Zeilen spielen für TPSign keine Bedeutung, sie erleichtern die Lesbarkeit durch den Benutzer.[8]

Andere Zeilen enthalten entweder Sektionsnamen oder Parameterwerte. Sektionsnamen werden immer in eckige Klammer eingeschlossen. Sektionsnamen erlauben beliebige Schreibweise, um die eckige Klammer dürfen nur Leerzeichen stehen.

Parameter und ihre Werte werden durch das Gleichheitszeichen getrennt. Es müssen beide vorhanden sein. Parameternamen erlauben beliebige Schreibweise, bestimmte Parameter akzeptieren auch Werte in beliebiger Schreibweise, jedoch nicht generell.

Alle Parameterwerte gehören der Sektion an, die zuvor mit dem Namen eingeleitet wurde. Die Reihenfolge, in der die Parameter auftreten, hat keine Bedeutung.

2.3.2 Hauptsektion

```
[*MAIN]
```

Die Hauptsektion hat den Namen '***main**' in beliebiger Schreibweise. Großschreibung wird bevorzugt um die Bedeutung hervorzuheben: Die Hautsekton konfiguriert den Teleporter generell.

Benutzerparameter

```
user        = all
```

Der Parameter '**user**' gibt an, wer den Teleporter benutzen kann, d.h. damit reisen darf. Der Wert ist eine Liste von Stringflags, verbunden via '**+**'. Es sind Stringflags '**owner**', '**!owner**', '**group**', '**!group**', '**all**' und '**!all**' erlaubt.

Sind die Flags '**all**', '**!all**' angegeben, werden andere ignoriert. Im ersten Fall kann jeder den Teleporter nutzen, im zweiten – niemand. Die Flags '**owner**', '**!owner**' erlauben oder verbieten, dem Gerätebesitzer, mit dem Teleporter zu reisen. Die Flags '**group**' und '**!group**' erlauben oder verbieten jeder Person aus der Objektgruppe, den Teleporter zu nutzen.

[8] Aus Performancegründen wird es empfohlen, auf Leerzeilen und reine Kommentarzeilen zu verzichten.

In der originalen Konfiguration 2.4 darf jeder den TPSign benutzen. Der Wert '**group+!owner**' würde dagegen bedeuten: Jeder aus der Objektgruppe kann mit dem Teleporter reisen, der Besitzer jedoch nicht.

Bildparameter

```
image       = 09BB628B-F97A-44FD-9503-6E4EC62DE59E
side        = all
```

Der Parameter '**image**' gibt das Defaultbild an, dieses wird angezeigt, wenn die Zielsektion kein Bild angibt. Erlaubt ist entweder eine UUID des Bildes, oder ein Bild im Inventar des Systemprims. Dann muss es Fullperm sein, damit seine UUID ermittelt werden kann.

Der Parameter '**side**' gibt die Primseite zum Anzeigen des Bildes an. Mögliche Werte sind '**all**', '**none**' und eine Zahl zwischen 0 und 9. Der Wert '**all**' bedeutet: Jede Primseite zeigt das Zielbild an, der Wert '**none**' unterdruckt die Bildanzeige. Ist dagegen eine Zahl angegeben, ist dies die LSL Nummer der Primseite für die Bildanzeige.

Titelparameter

```
title       = %n\n...touch to visit
color       = lavender
fadeout     = 0
```

Der Parameter '**title**' legt den Titel fest (den Hovertext.) Der Wert kann Platzhalter enthalten:

- Die Platzhalter '**\n**' und '**\t**' werden durch einen Zeilenumbruch und Tabulatorzeichen ersetzt.
- Die Platzhalter '**%n**' und '**%s**' stehen für den Namen des aktuell angezeigten Ziels, so wie Namen der LM oder der Sim, mit der das Ziel definiert ist.
- Die Platzhalter '**%p**' und '**%c**' werden durch die Nummer des angezeigten Ziels in der Zielliste ersetzt (1 steht fürs erste Ziel) und durch die Anzahl Ziele in der Liste.

Zum Verbergen des Titels kann man den Parameter auslassen oder '**-**' als Wert angeben.

Der Parameter '**color**' gibt die Titelfarbe an. Der Parameter kann auch als '**colour**' angegeben werden. Der Farbwert wird grundsätzlich ein LSL Vektor. Einige gebräuchliche Farben können aber auch mittels ihren Namen angegeben werden. Die

erkennbaren Farbnamen sind in der RTS Dokumentation angegeben. Der Farbname beachtet die Schreibweise nicht, '**White**' bedeutet dasselbe wie '**WHITE**' oder '**white**'.

Der Parameter '**fadeout**' legt die Anzeigedauer in Sekunden fest, bevor der Titel ausgeblendet wird. Der Timer startet immer, wenn der Titeltext sich ändert. 0 als Wert bedeutet: Kein Ausblenden.

Teleportmodi

```
mode        = RLV
fail        = LM
return      = LM
delay       = 10
```

Der Wert von Parametern '**mode**', '**fail**' und '**return**' ist eine Kombination von Stringflags '**RLV**', '**LM**', '**N**' und '**OFF**'. Die Flags können via '**+**' kombiniert werden und bestimmen das Verhalten (die Arbeitsweise) des Teleporters. Die Flags wirken additiv, die Einstellung '**RLV+OFF**' bedeutet dasselbe wie '**RLV**'.

Ist der Flag '**RLV**' gesetzt, spricht der Teleporter die RLV Schnittstelle des Benutzerviewers an. Der Flag '**LM**' erlaubt die Herausgabe der LM bzw SLURL zum Ziel, der Teleporter wird zum Landmarkverteiler. Die Flags '**N**' und '**OFF**' deaktivieren den entsprechenden Modus.

Der Parameter '**mode**' bestimmt, ob der Teleportprozess via RLV oder mittels Übergabe der LM bzw. SLURL zum Ziel auszuführen ist. Erlaubte Flags sind: '**RLV**', '**LM**', '**N**' und '**OFF**'. Ist das Flag '**LM**' angegeben, gilt der Teleport immer als erfolgt, da der Agent die übergebene LM bzw. SLURL nutzen kann.

Der Parameter '**fail**' gibt an, ob die LM bzw. SLURL zum Ziel an den Agenten übergeben wird, falls der Teleport via RLV von ihm abgelehnt wurde. Erlaubt sind Flags '**LM**', '**N**' und '**OFF**'.

Der Parameter '**return**' gibt an, ob eine SLURL zum Standort des Teleporters an den Agenten übergeben wird, falls er teleportiert wurde. Erlaubt sind Flags '**LM**', '**N**' und '**OFF**'.

Der Parameter '**delay**' bestimmt, wie lange auf die Rückmeldung des Realay gewartet wird, bis das Timeout festgestellt ist und der Modus '**fail**' in Kraft tritt. Die Zeit ist in Sekunden angegeben, das Minimum ist 5 Sekunden. In diese Zeit fallen sowohl lagbedingte Verzögerungen, als auch die Reaktionszeit des Benutzers, auf die Anfrage seines Relays zu reagieren, Abbildung 2.3.

Teleportrelay

```
norelay     = TP Relay (r upper arm)
```

Der Parameter '**norelay**' gibt die Datei an, die übergeben wird, falls keine Antwort vom Relay des Benutzers kam. Die Datei ist mit 'Teleportrelay' in der Abbildung 2.3 bezeichnet (das Timeout-Zweig.)

Auf diese Weise lässt sich etwa eine Notekarte mit weiteren Informationen übergeben. TPSign ist auf die Übergabe des Teleportrelays eingestellt, da damit der Teleporter auch ohne RLV-fähigen Viewer benutzbar ist.

Offsetparameter

```
offset = -3          # entspricht...
offset = <-3, 0, 0>
```

Der Parameter '**offset**' gibt die relative Position des Rückkehrpunktes zum Teleporter an, das ist der Punkt wo die zurückkehrende Avatare gerezzt werden. Die Position kann mittels einer Zahl oder eines Vektors angegeben werden. Die Angabe per Einzelzahl entspricht der Angabe via Vektor, wo die Zahl als x Komponente gilt. Beide Definitionen im obigen Beispiel sind daher identisch.

Was bedeutet dieser Wert? Ist der Offset null, ist die Rückkehrposition die Position des Teleporters selbst, dann werden Avatare, welche die Rückkehr-SLURL benutzen, am Teleporters gerezzt. Manchmal ist das ungünstig:

Ist der Teleporter etwa am Tisch oder an der Wand montiert, kann SL die Rezzposition unter dem Tisch oder hinter der Wand auswählen. Die Offseteinstellung kann den Rückkehrpunkt zur Seite verschieben zu einem Ort das sicher zu rezzen ist.

Hinweis: Die Position eines Avatars ist Zentrum seiner Boundingbox, dieser Punkt ist in etwa 1 Meter über dem Boden.

Der Offsetvektor ist immer ein Vektor in der globalen xy Ebene um den Systemprim. Ist es nicht rotiert, zielt der Vektor **<3, 0, 0>** auf den Punkt 3m richtungs der lokalen x Achse des Prims. Wird das Systemprim um die z Achse rotiert, folgt der Offsetvektor dieser Rotation, ignoriert aber Rotationen um die x und y Achsen.

Auf diese Weise kann man einen Teleporter an der Wand so konfigurieren, dass der Rückkehrpunkt 3 Meter vor dem Teleporter liegt. Wird der Teleporter an jeder anderen Wand platziert, liegt der Rückkehrpunkt immer noch 3m vor ihm. Wird der Teleporter aber flach auf den Boden oder Tisch gelegt, verschiebt sich der Rückkehrpunkt nicht über den Teleporter, sondern bleibt 3m seitlich von ihm.

FastTP Modus

```
fasttp      = OFF
```

Dieser Parameter ist aus Kompatibilitätsgründen in der TPSign Konfiguration enthalten. Der Parameterwert '**Y**' und '**ON**' schaltet den FastTP Modus ein, und der Wert '**N**' und '**OFF**' schaltet ihn aus. Da der TPSign für nur ein Ziel ausgelegt ist, hat hier der FastTP Modus keine Bedeutung.

Erlaubt der Teleporter die Auswahl aus mehreren Zielen,[9] lässt sich dieser Modus aktivieren. Ist der FastTP Modus an, erfolgt bei einer Zielauswahl sofort der Teleport zum ausgewählten Ziel. Ist der Modus ausgeschaltet, kann man durch die Ziele blättern, den Teleport muss man dann zusätzlich starten.

Zielaktualisierung

```
refresh     = 24
```

Zum Teleportieren benötigt TPSign die globale Koordinate der Zielposition. Sims bewegen sich manchmal, was diesen Wert verändert. Die globale Zielkoordinate muss erneut aufgelöst werden, das bedeutet Serverlast und Verzögerungen.

Um die Server zu entlasten, werden die aufgelöste Koordinaten zwischengespeichert. Der Parameter '**refresh**' gibt an, wie lange gespeicherte Daten gültig sind. Die Angabe ist in Stunden, der Minimalwert ist 6. Das Wiederauflösen findet dabei nicht permanent statt, sondern erst beim Teleport auf eine veraltete Position.

Es empfiehlt sich ein Wert von einem Tag bis einer Woche zu nehmen (24 bis 168 Stunden.) Auch lassen sich die Zielpositionen via 'refresh' Button im Besitzermenü manuell als veraltet kennzeichnen.

Nachrichtformat

```
format = APP
```

Der Parameter '**format**' legt das Format von übergebenen SLURLs fest. Hier kann man einen der vier Werte angeben: '**MAPS**', '**SLURL**', '**APP**' und Freitext.

'**SLURL**': Versende eine *SLURL*.[10] Sie ist klassisch, alle Viewer verstehen den Link und öffnen das Landmarkenfenster, wenn der Link im Chatverlauf angeklickt ist.

[9] Der TPVendor, der im Abschnitt 2.4 beschrieben ist, ist ein solcher Teleporter
[10] Ein Link wie http://slurl.com/secondlife/Endora/123/234/345

'**MAPS**': Versendet eine moderne *MapURL*.[11] Sie wurde ab der Version 1.23 des SL Viewers eingeführt. Ältere Viewer öffnen den Webbrowser.

APP: Versendet einen *Teleportlink*.[12] Dieser String löst beim Klick im Chatverlauf einen Teleport ohne eines geöffneten Landmarkenfenster aus.

Der Freitext ist ein String mit Platzhaltern `%s`, `%x`, `%y`, `%z`, die durch den Simnamen und die lokale Koordinaten ersetzt werden. Zum Beispiel produziert der Format '`%s (%x, %y, %z)`' einen String wie 'Endora (123, 234, 345)'.

Hinweis: Der Freitext kann die Flags '**MAPS**', '**SLURL**' und '**APP**' emulieren. Zum Beispiel erzeugt der Formatstring '`secondlife:///app/teleport/%s/%x/%y/%z`' denselben Teleportlink wie das Flag '**APP**', benötigt jedoch mehr Prozessorressourcen zum Generieren des Strings.

Nachrichtparameter

```
msg:url     = This link brings you at target location:
msg:lm      = This landmark brings you at target location...
msg:return  = This link brings you back:
msg:norelay = Please, use this item in order to use me...
```

Die Parameter '**msg:...**' erlauben ebenfalls die Angabe von Textplatzhaltern, Abschnitt 2.3.2 und speichern Textnachrichten die beim Übergeben von Items oder SLURLs mitversendet werden.

Der Parameter '**msg:url**' gibt den Text an, der die Ziel-SLURL einleitet. Die IM an den Reisenden besteht aus diesem Text, gefolgt von der SLURL.

Der Parameter '**msg:lm**' gibt den Text an, der beim Übergeben der Ziel-LM als IM versendet wird. Der Reisende bekommt die LM und die IM getrennt.

Der Parameter '**msg:return**' gibt den Text an, der die Rückkehr-SLURL einleitet.

Der Parameter '**msg:norelay**' gibt den Text an, der beim Übergeben des norelay-Elements versendet wird.

2.3.3 Zielsections

Eine Zielsektion besteht aus nur drei Einträgen: Der *Zielname*, das *Zielbild* und die *Zielangabe*. Da TPSign keine Möglichkeit vorsieht, Ziele auszuwählen, wird permanent nur das erste Ziel ausgewählt, deshalb macht für TPSign nur eine Zielsektion Sinn. Definierbar sind jedoch bis zu 80 Zielsektions.

[11] Ein Link wie http://maps.secondlife.com/secondlife/Endora/123/234/345
[12] Ein String wie secondlife:///app/teleport/Endora/123/234/345

Zielname

```
[Welcome to Endora]
```

Der Zielname ist der Sektionsname. Er muss mit einem Buchstaben oder Unterstrich beginnen und kann aus mehreren Wörtern bestehen. Er muss aber eindeutig und höchstens 32 Zeichen lang sein. Der Zielname kann via Plazhalter '**%n**' in den Titel (Hovertext) und versendete Nachrichten eingebracht werden.

Zielbild

```
image       = 8cee492c-917e-341d-b3a9-63d83ed44d34
```

Das Zielbild wird stellvertretend für das Ziel angezeigt. Der Parameter '**image**' ist optional. Ist er ausgelassen, wird das Defaultbild aus der Hauptsektion verwendet.

Als Parameterwert kann sowohl die UUID des Bildes angegeben werden oder Name eines Bildes im Objektinventar des Systemprims. Dann muss es dort fullperm vorliegen, sonst kann seine UUID nicht ermittelt werden.

Zielangabe

```
target      = Endora/127/169/42
```

Der Parameter '**target**' ist obligatorisch und definiert die Zielposition. Die Angabe kann via LM oder einer SLURL erfolgen.

Ist das Ziel via LM angegeben, muss sie im Objektinventar vorliegen. In diesem Fall wird die LM an den Reisenden übergeben, falls LM-Übergabe aktiv ist, Abschnitt 2.3.2. Dadurch kann er das Ziel später noch mal besuchen, ohne den Teleportlog zu durchsuchen.

Ist das Ziel als SLURL angegeben, wird das Inventar des Reisenden nicht durch Objekte gefüllt, die er nicht mehr wieder braucht, das Wiederbesuchen ist nur über den Log oder manuell gezogenen LM möglich. Die SLURL kann auf drei Arten angegeben werden:

- SLURL: http://slurl.com/secondlife/Endora/127/169/42
- MapURL: http://maps.secondlife.com/secondlife/Endora/127/169/42
- RawURL: Endora/127/169/42

RawURL beginnt mit dem Simnamen und ist daher universell. **Hinweis**: In der RTS Dokumentation kann diese Form 'Stripped SLURL' bezeichnet werden, die Bezeichnung 'RawURL' wurde später eingeführt.

2.4 Erweiterung

TPSign ist ein RTS Teleporter, benutzt aber nicht alle Möglichkeiten von RTS: TPSign wurde ursprünglich als ein Teleporter für nur ein Teleportziel entwickelt. RTS kann mit einer Liste von Ziele arbeiten, TPSign bietet jedoch traditionsbedingt keine Möglichkeit, das Teleportziel zu wählen.

In diesem Abschnitt wird der Teleporter um diese Möglichkeit erweitert. Wir müssen dafür keine Skripte schreiben oder verändern. Alles was wir brauchen ist, richtig benannte Prims zum Teleporter hinzu zu linken. Das Ergebnis (aber nicht das Ziel der Erweiterung) wird ein Teleporter sein, der unter dem Namen TPVendor im RTS Paket liegt, Abbildung 2.5.

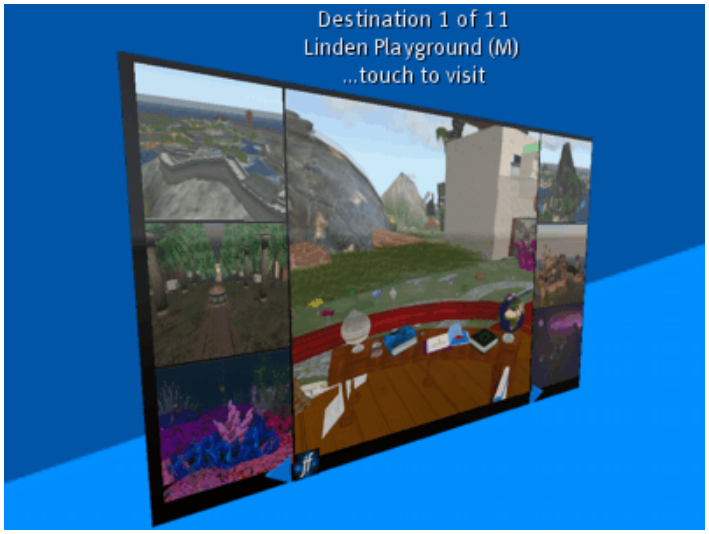

Figure 2.5: TPVendor

Der Abschnitt richtet sich an Käufer des TPSign – da man gridweit teleportieren kann, braucht man bald einen Teleporter für mehr als nur ein Ziel. Aber auch an Besitzer des RTS Pakets, als Demonstration von RTS und als Tutorial für die Arbeit damit.

2.4.1 TPSign Architektur

Zuerst betrachten wir kurz den Aufbau und Arbeitsweise von TPSign. Das mag es helfen, zu verstehen, wie TPSign funktioniert. Detailierte Inforrmation findet sich in der RTS Dokumentation, hier finden wir lediglich einen Überblick.

Dateien

Im TPSign sind einige Dateien installiert.

Skript '**.core**': Kernskript. Er kontrolliert die anderen Skripte, führt den Teleport aus und initialisiert den Versand von Nachrichten und Dateien. Er präsentiert außerdem das Besitzermenü und führt eine Reihe von Kernservices.

Skript '**.config**': Konfigurationsskript. Er liest die '**config**' Notekarte und verteilt die gelesenen Daten an andere Skripte. Andere Skripte haben auf die Konfigurationsnotekarte kein Zugriff, das vereinfacht deren Code.

Skript '**.sender**': Senderskript. Er versendet Dateien wie LMs oder das Relay, so wie Nachrichten per IM an die Benutzer, aber auch Fehlermeldungen, die während der Konfiguration und im Betrieb entstehen.

Skript '**.targets**': Zielliste, kann bis zu 80 Ziele aufnehmen, verwaltet die in der Konfiguration angegebene Ziele: Speichert Daten aus der Konfiguration, löst erforderliche Zielkoordinaten auf, liefert Daten für ausgewähltes Ziel.

Objekt '**TP Relay (r upper arm)**': Das Teleportrelay, wie in der Konfiguration angegeben, Parameter '**`norelay`**', Abschnitt 2.3.2.

Kernservices

In RTS wird die Funktionalität des Teleporters in einzelne Kernservices unterteilt. Sie werden durch den Kernskript ausgeführt, sind aber übertragbar. Folgende Kernservices kommen im TPSign zum Einsatz:

Das *Textservice* ist dafür zuständig, den Titeltext über dem Teleporter anzuzeigen.

Das *Nachrichtenservice* versendet Fehlermeldungen an den Besitzer via Senderskripts.

Das *Imageservice* stellt das Zielbild dar.

Das *Touchservice* startet den Teleport beim Anklicken des Teleporters.

Das *Menüservice* öffnet das Besitzermenü beim Langklick auf den Teleporter.

2.4.2 Aufbau

Zurück zur ursprünglichen Aufgabe. Der Teleporter wird ein Linkset, wobei die Prims grob vier Aufgaben übernehmen. Die *Basis* ist ein Prim wo alle Skripte installiert sind, und das tatsächlich der unveränderte TPSign ist.

Die *Vorschaubilder* und *Skrollbuttons* erlauben es, Teleportziele auszuwählen. RTS bietet dafür mehrere Möglichkeiten, sir benutzen diese zwei.

Das *Menübutton* macht es einfacher, das Besitzermenü zu öffnen, als mit dem Langklick. Schließlich hält das Hintergrundprim die gesamte Konstruktion zusammen, ist jedoch für den Betrieb des Teleporters nicht erforderlich.

Basis

Als erstes benötigen wir den Basisteleporter: TPSign wird gerezzt und auf die Größe (0.01, 1.52, 1.52) gebracht, Abbildung 2.6. Die 2cm Vergrößerung gehen auf den Abstand zwischen den Vorschauprims, die im nächsten Schritt folgen. Das Prim benennen wir gleich um – zu '*TPVendor*'.

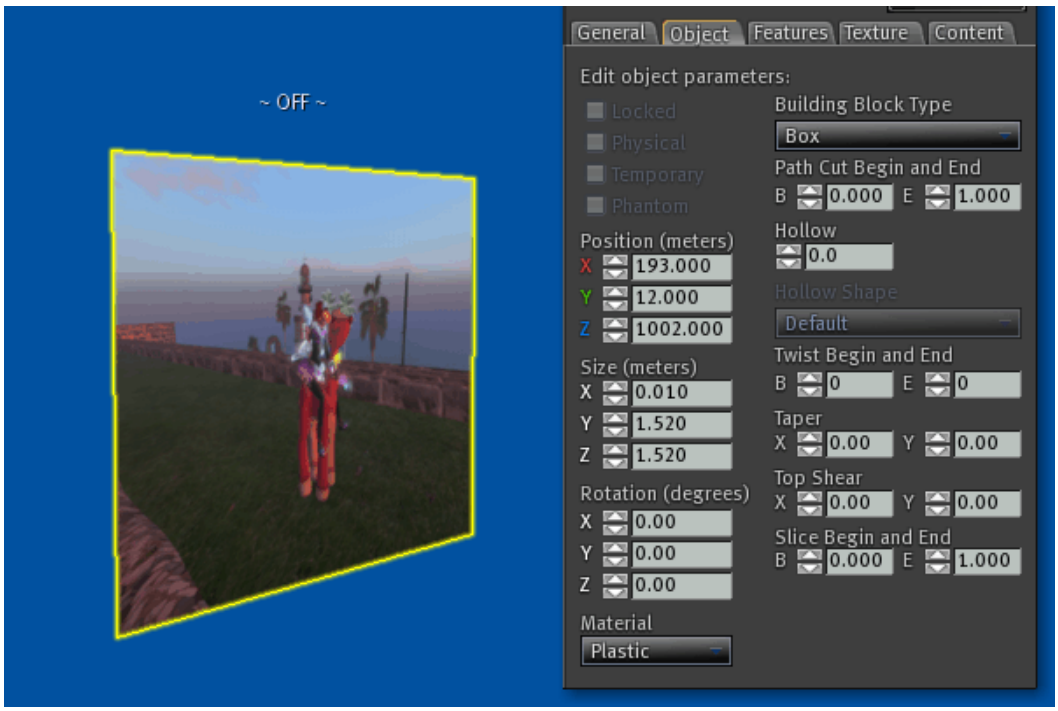

Figure 2.6: Rezzen der Basis

Alternativ, konnten wir eine Primbox in dieser Größe erstellen und die entsprechende Dateien installieren, Abschnitt 2.4.1.

Vorschauprims

RTS unterstützt bis zu 20 Vorschaubilder, wir nehmen nur 6. Auf den Prims werden die Zielbilder angezeigt, Klick darauf wählt das angezeigte Ziel.

Wir erstellen eine Primbox von der Größe (0.01, 0.5, 0.5) und positionieren sie neben der Basis: x und z Werte werden von der Basis übernommen, der y Wert um 1.20m vergrößert. Diese Primbox wird nun nach oben und nach unten kopiert: Bearbeiten, die Umschalttaste gedruckt halten und die Primbox am blauen Pfeil ziehen.

Beide Kopien werden 51cm über und unter dem Original platziert. Alle drei Primboxen werden ausgewählt und auf dieselbe Weise zur Seite kopiert: Die drei neuen Kopien werden auf der anderen Seite neben der Basis platziert.

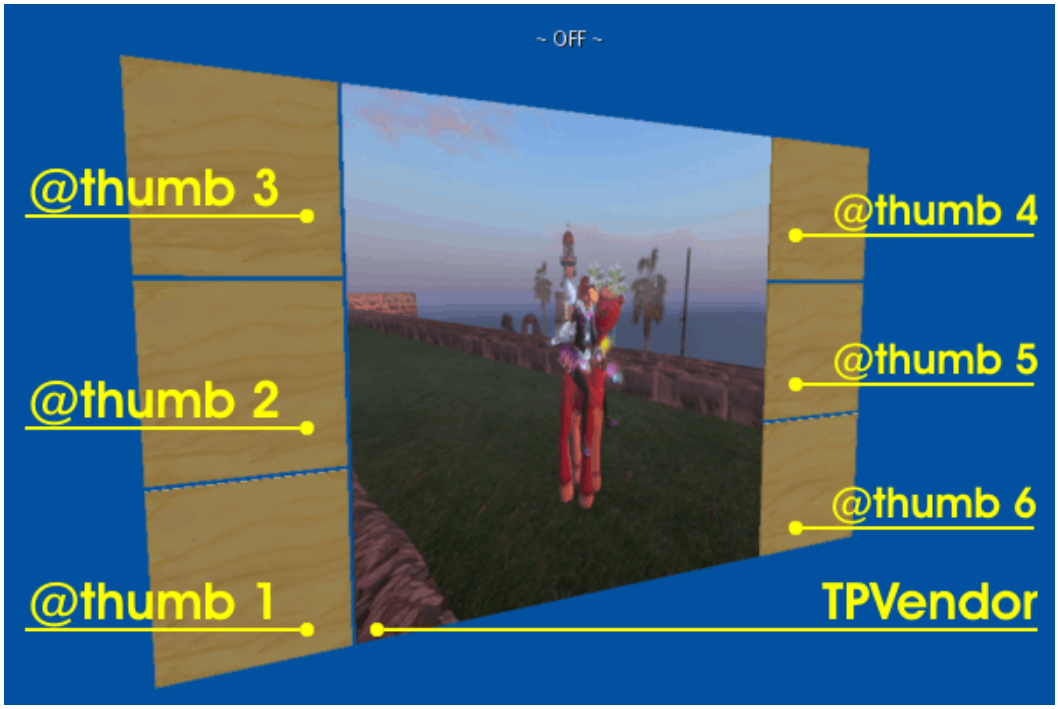

Figure 2.7: Vorschaubilder

Damit die Prims als Vorschauprims erkannt werden, müssen wir sie speziell benennen: Der Primname beginnt mit '`@thumb`' gefolgt mit einer laufenden Nummer. Die Nummer bestimmt die Anzeigereihenfolge der Ziele, Abildung 2.7.

Scrollbuttons

Scrollbuttons sind zwei Prims, welche die Zielauswahl um eine bestimmte Anzahl Ziele verschieben. Eine Prismenform kommt einem Pfeil am nächsten und benötigt keine Textur.

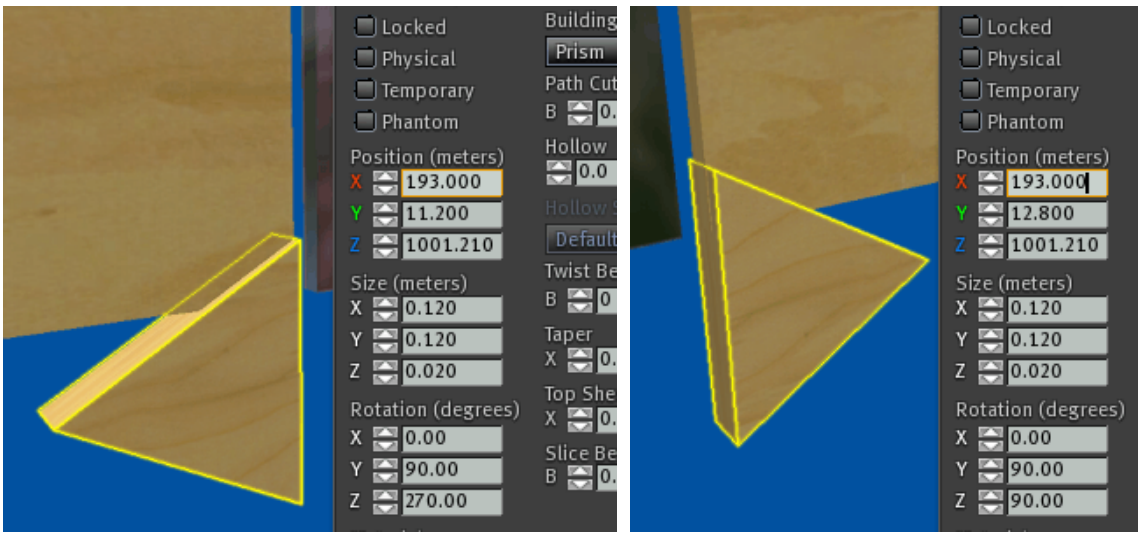

Figure 2.8: Scrollbuttons

Wir erstellen ein Prisma der Größe (0.12, 0.12, 0.02) und positionieren es bündig unterhalb des linken Vorschauprims. Das Prisma wird dann nach rechts kopiert und bündig unter dem rechten Vorschauprim platziert, Abbildung 2.8.

Damit die Prims als Scrollbuttons erkannt werden können, müssen sie ebenfalls speziell benannt werden. Zum Vorausblättern muss der Primname mit '**next**' anfangen, gefolgt von der Anzahl Stellen um die geblättert wird. Zum Zurückblättern muss der Primname mit '**prev**' anfangen, gefolgt von der Stellenzahl.

Die Vorschauprims und das Zielprim zeigen zusammen genau 7 Ziele an. Daher ist es sinnvoll, beim Klick auf Scrollbuttons um genau 7 Stellen zu blättern. Die Buttons sind daher mit '**prev7**' und '**next7**' zu benennen. Beide Prismen erhalten später eine blanke Textur und blaue Farbe.

Menübutton

Der Menübutton ist ein Prim, das einerseits unser Logo enthält, und andrerseits beim Anklicken das Besitzermenü aufruft. Dadurch spart man den Langklick. Man könnte ein neues Prim erstellen, am einfachsten kopiert man aber den rechten Scrollbutton, ändert die Form zu einer Box, korrigiert die Größe und richtet bündig unter der Basis aus, Abbildung 2.9.

Sie ahnen schon: Das Prim muss einen speziellen Namen haben, damit es als Menubutton funktioniert. Der Primname ist in diesem Fall genau festgelegt: '**.menu**' (der Punkt ist kein Tippfehler.)

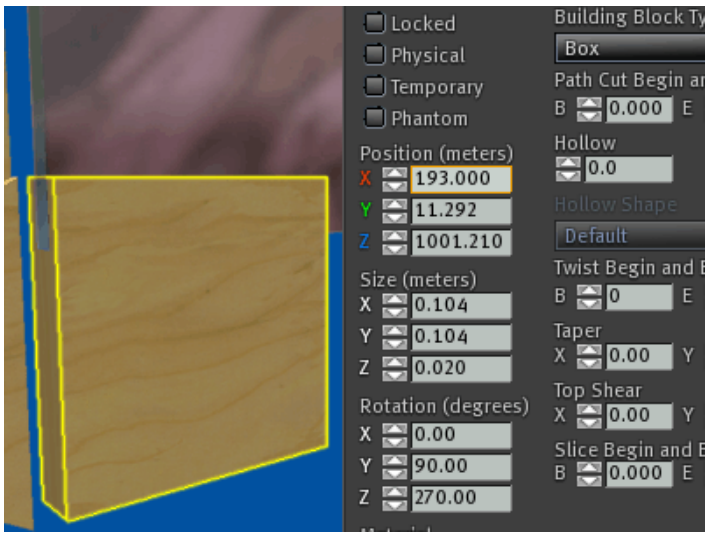

Figure 2.9: Menubutton

Abschluss

Wir sind fast fertig. Es fehlt nur das Hintergrundrim. Dabei ist nichts Besonderes zu beachten, außer das Prim muss hinter allen Prims passend ausgerichtet werden. Schließlich werden alle Prims so verlinkt, dass die Basis das Rootprim wird. In einem Rutsch können wir nun den Glanz und das Leuchten zuweisen. Das Ergebnis ist in 2.10 abgebildet.

2.4.3 Konfiguration

Der Aufbau ist fertig. Es fehlt noch die Konfigurationserweiterung. Dazu bearbeiten wir den Teleporter und die Notekarte '**config**' darin.

Hier wird lediglich der Titel geändert (um die Mehrzielfähigkeit zu reflektieren) so wie die eine Zielsection durch eine Liste von Zielsections ersetzt – einmal für jedes Teleportziel. Wir können hier bis zu 80 davon registrieren. TPVendor kommt mit 11 registrierten Beispielzielen, Abbildung 2.11.

Vergleichen Sie bitte mit Konfiguration von TPSign, Abildung 2.4. Wie man sehen kann, in der Hauptsektion wurde nur der Titel verändert und es sind mehr Zielsections. Zielnamen enthalten das Rating der Sim (G, M, A) in den Klammern. Der Zielname wird über den Platzhalter '**%n**' in den Titeltext integriert, das erlaubt es, im Betrieb, noch vor dem Teleport die Rating des Zielorts zu erkennen.

Zielbilder sind via UUID angegeben. Anschließendes Kommentarzeichen erlaubt es, den Bildnamen anzugeben. Der Bildname ist nicht für den Teleporter bestimmt, er hilft später, das verwendete Bild im Inventar wiederzufinden.

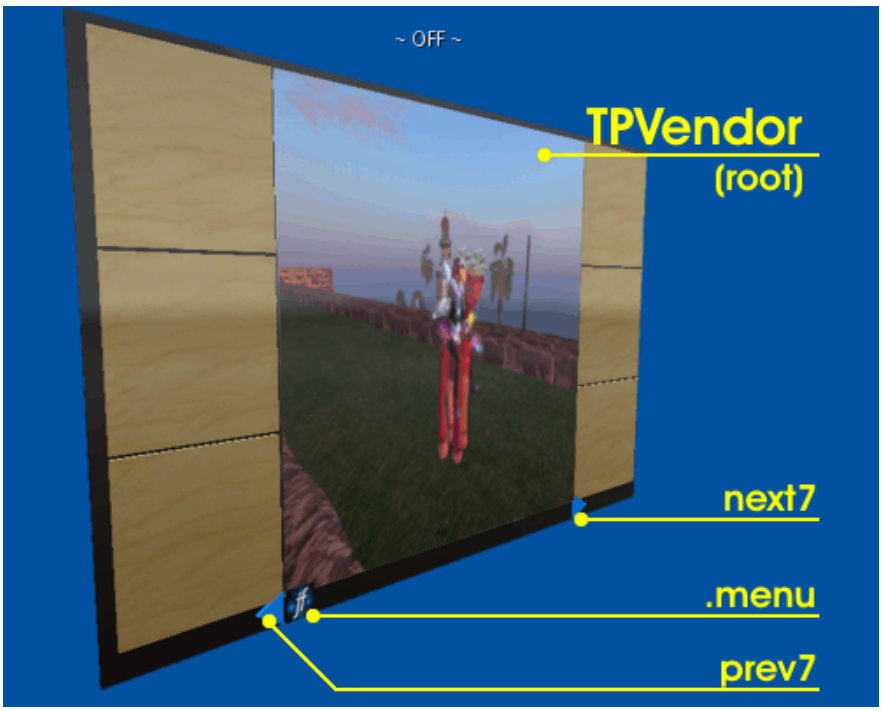

Figure 2.10: Abschluss

Zielpositionen wurden via RawURL. Dadurch braucht man keine Landmarken anfertigen um die Ziele vorzubereiten, man kann auch den Teleportverlauf benutzen oder Angaben der Sim und die Positionen von den Screenshots ablesen.

Nachdem die Konfiguraton erweitert und abgespeichert wurde, muss man den Teleporter starten: Den Menübutton anklicken, '**reset**'-Button im geöffneten Menü. Nach Ablauf der Konfiguration muss der Teleporter via erneut geöffneten Besitzermenü starten – der Teleporter ist einsatzbereit, wie in 2.5 abgebildet.

2.4.4 Erweiterte Architektur

Es wurden im Originalteleporter nichts an den Skripte geändert. Warum funktionieren die Änderungen? Die Antwort liegt in der Architektur und Schnittstellen von RTS. Dieser Abschnitt soll ihre Wechselwirkung erklären, im Kontext des TPVendors. Für weitere Informationen schlagen Sie bitte die RTS Dokumentation nach.

Die Zielliste (Skript '**.targets**') verwaltet die in der Konfiguration angegebene Ziele. Wird ein Ziel (auf beliebige Weise) ausgewählt, liefert die Zielliste eine Reihe von Zieldaten, unter anderem das Zielbild und Bilder der Nachbarziele.

Durch Hinzufügen und Benennen von Prims werden zwei weitere Kernservices aktiviert, das *Thumbservice* und das *Buttonservice*.

```
# TPVendor Konfiguration
[*MAIN]
title       = Destination %p of %c\n%n\n...touch to visit
# --- Der Rest wie für TPSign --------------------------

# Zielliste
[Linden Playground (M)]
image       = 1b5e8133-3fb8-5488-c498-e63ec41b2010 # Da Boom (143, 148, 41)
target      = Da Boom/128/128/35

[Protected Ocean (G)]
image       = 16c62e5b-0b9c-9d78-1aba-e1f45af68bdb # Pravatch (230, 229, 2)
target      = Pravatch/221/227/3

[MYST Island (G)]
image       = 5a6b672a-6c6a-d2a1-a020-04217cdaf0da # Age of Myst (171, 46, 33)
target      = Age of Myst/175/14/24

[Great Wall Of China (G)]
image       = 10594b0c-2ae4-3f7b-052b-a77a098fefcd # China Sichuan (130, 10, 24)
target      = China Sichuan/133/16/24

[Museum of Illusions & Magic (G)]
image       = 76999bdb-11ba-c08d-4d5a-24ea1bed073a # AmazingIllusionMuseumEI
target      = Enchantment Island/59/120/45

#Dieses Ziel war beim TPSign angegeben
[Welcome to Endora (A)]
image       = 8cee492c-917e-341d-b3a9-63d83ed44d34
target      = Endora/127/169/42

[The Grand Canyon (M)]
image       = d835568f-fb32-5fe4-2ba1-99191261eced # Grand Canyon (106, 173, 110)
target      = Grand Canyon/89/186/110

[Happy Mood (G)]
image       = 525ea54c-49dd-bc1d-327e-c64cd90610db # HappyMood (107, 144, 79)
target      = HappyMood/109/144/80

# --- 3 weitere Ziele folgen ---------------------------
```

Figure 2.11: Konfiguration von TPVendor

Durch das Namenspräfix '**@thumb**' werden die Vorschauprims als solche gekennzeichnet und werden durch das *Thumbservice* erfasst. Wird ein Ziel ausgewählt, empfängt das Service die Vorschaubilder und zeigt sie auf den Vorschauprims an. Beim Klick eines davon produziert das Service eine Steuernachricht, welche die Zielauswahl derart verschiebt, dass genau das angeklickte Vorschauziel ausgewählt wird.

Die Prims mit Namen '**prev7**', '**next7**' und '**.menu**' gelten als Buttons und werden durch das *Buttonservice* erfasst. Das Service produziert beim Klick darauf eine Steuernachricht, die den Buttonnamen überträgt.

Beginnt der Buttonname mit einem Punkt, ist die Steuernachricht für den Kernskript bestimmt, im Falle des Menübuttons (Primname '**.menu**') ist die Steuernachricht genau die zum Öffnen des Besitzermenüs.

Beginnt der Buttonname mit einem Buchstaben, ist die Steuernachricht für die Zielliste bestimmt. Das Namenspräfix '**prev**' bzw. '**next**' veranlasst die Zielliste, die Zielauswahl zu verschieben, die nach dem Präfix kommende Zahl gibt dabei Anzahl Stellen. Die Scrollbuttons verschieben so die Zielauswahl um 7 Stellen. Der Thumbservice versendet ähnliche Steuernachrichten, um die Zielauswahl um 1, 2 oder 3 Stellen vor und zurück zu verschieben.

TPVendor benutzt in der vorgestellten Ausführung also nur eine Auswahlmöglichkeit. Die Zielliste bietet aber auch Weitere: Die absolute Positionsangabe in der Zielliste (Buttonpräfix '**pos**'), Auswahl des ersten und letzten Ziels (Buttons '**first**', '**last**'), Zufallsauswahl (Button '**random**') und eine Textsuche.

Das wäre alles. Ich wünsche Ihnen viel Spaß und Erfolg beim Experimentieren mit dem Teleporter und dem RTS.