TREMULOUS 1.1.0

17 marzo 2006

Indice

1 Introduzione

Tremulous è uno sparatutto in prima persona composto da due team avversari, Umani e Alieni. Entrambi I team possono costruire delle strutture e punti di ingresso (spawn), che sono importantissimi per la vittoria.

Lo scopo di Tremulous e quello di eliminare il team avversario e tutti i loro spawn.

I due team sono fondamentalmente molto diversi. Gli alieni sono basati su classi; con due calssi inizialmente disponibili: L'alieno costruttore, il Granger, e il Dretch, l'alieno offensivo più debole.

Gli alieni ricevono Punti Evo per l'uccisione dei loro nemici che possono essere usati per evolversi in categorie più forti, esse sono capaci di grandi manovre delle caratteristiche più varie.

In maniera analoga gli aggiornamenti della squadra umana sono basati, su credi ricevuti per le uccisioni che possono essere scambiate da una struttura l'armoury (armeria) per nuove armi, armature ed apparecchiature.

Due di questi aggiornamenti sono disponibili gratis: il rifle (mitragliatore di base) e il Construction kit un corredo per la costruzione, delle strutture.

Durante la partita, ogni squadra passa per tre diverse fasi di sviluppo (Livelli). Questi livelli sono sbloccati raggiungendo un determinato numero di uccisioni sul totale dell'intera squadra.

Ogni nuovo livello sblocca nuove classi di aggiornamenti (per gli umani) o evoluzione(per gli alieni) che comprendono anche la possibilità di costruire nuove strutture.

Prima una squadra raggiunge passa di livello maggiore è la probabilità vittoria.

2 Gioco

2.1 Alieni

Due classi sono disponibili giocando con la squadra aliena: il Dretch ed il Granger. Una volta uccisioni gli avversari, potete usare i vostri frags guadagnati per evolversi in classi maggiori utilizzando il tasto STRUTTURA/EVOLVI (**predefinito Q** N.d.T.). La squadra aliena principalmente è limitata agli attacchi di prossimità e deve usare lo furtività e la velocità per sconfiggere gli esseri umani che al contrario posseggono armi di lunga gittata. Tutti gli alieni rigenerano automaticamente la salute in modo lento.

Granger

Costo: 0 Stage: 1

Abilità	Controlli
Costruisce	ATTACCO PRIMARIO
Distruggi struttura	DECOSTRUISCE STRUTTURA (solo strutture aliene)

Il *Granger* è il costruttore del team alieno. Con l'ATTACCO PRIMARIO accede al menu contenente le strutture costruibili. Dopo averla selezionata la struttura appare luminescente. quando diventa verde si può riusare l'ATTACCO PRIMARIO per piazzarla. La fluorescenza appare rossa quando non è possibile piazzare la struttura in un determinato post. per cancellare il posizionamento della struttura premere ATTACCO SECONDARIO. Per rimuovere una struttura piazzata premere DECOSTRUISCI STRUTTURA (**predefinito E** N.d.T.). Dopo aver costruito o decostruito apparirà nell'angolo in basso a destra un timer. Finche il tempo non sarà scaduto non si possono costruire o decostruire altre strutture.

Advanced Granger

Costo: 0 Stage: 2

Abilità	Controlli
Costruisce	ATTACCO PRIMARIO
Snippa	ATTACCO SECONDARIO
Proiettile a parabola	ATTIVA UPGRADE
Distruggi struttura	DECOSTRUISCE STRUTTURA (solo strutture aliene)
Arrampicamuri	ABBASSATI

Il *Granger Avanzato* diviene disponibile gratuitamente quando gli Alieni raggiungono il Livello 2. in più rispetto al *Granger*, il *Granger Avanzato* può muoversi più velocemente, saltare più in alto, camminare sui muri, attaccare con lo snippo o sputacchiare piccoli proiettili con traiettoria parabolica con l'ATTIVA UPGRADE.

Dretch

Costo: 0 Stage: 1

Abilità	Controlli
Morde	Toccando un umano
Arrampicamuri	ABBASSATI

Il *Dretch* ie la classe più debole tra gli alieni. Il suo solo attacco avviene tramite il contatto con il giocatore umano o le sue strutture difensive. L'ammontare del danno causato dipende da che tipo di armatura indossa l'avversario e da dove viene colpito, il colpo alla testa è il più dannoso (può risultare fatale se l'umano non ha armature N.d.T.). I *Dretch* possono anche arrampicarsi sui muri premendo il tasto ABBASSATI.

Basilisk

Costo: 1 Stage: 1

Abilità	Controlli
Slash	ATTACCO PRIMARIO
Afferrare	Toccando un umano
Arrampicamuri	ABBASSATI

Il *Basilisk* aattacca con l'ATTACCO PRIMARIO. Può anche Afferrare un giocatore umano trmite contatto a distanza ravvicinata. Questo congela gli umani sul posto, se sono provvisti di *Battlesuit*, restringe l'abilità nel girare. I *Basilisk* possono anche arrampicarsi sui muri premendo il tasto ABBASSATI.

Advanced Basilisk

Costo: 2 Stage: 2

Abilità	Controlli
Slash	ATTACCO PRIMARIO
Gas	ATTACCO SECONDARIO
Afferrare	Touch a human
Arrampicamuri	ABBASSATI

In più rispetto al *Basilisk,* il *Basilisk Avanzato* può spruzzare una nuvola di gas nocivo che disorienta l'umano e lo avvelena.L'umano equipaggiato di *Battlesuit* è immune al gas. Tale abilità è attivata tramite l'ATTACCO SECONDARIO.

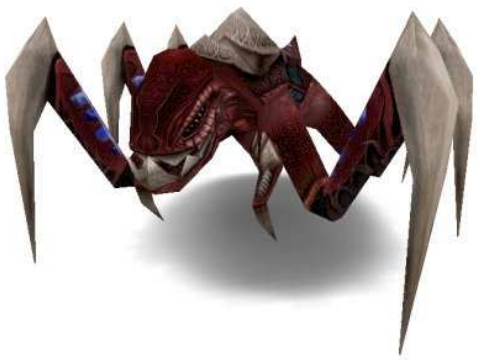
Marauder

Costo: 2 Stage: 1

Abilità	Controlli
Morde	ATTACCO PRIMARIO
Salto sui muri	Salta tra i muri premendo SALTA

Il *Marauder* attacca con l'ATTACCO PRIMARIO e ha l'bilità di rimbalzare tra i muri. per usare tale abilità premere SALTA in prossimità di un muro. Quando colpite la parete sarete respinti verso l'alto e nella dirzione opposta della parete. Finchè continuate a colpire le pareti continuerete il salto tra i muri.

Advanced Marauder



Costo: 3 Stage: 2

Abilità	Controlli
Morde	ATTACCO PRIMARIO
Fulmina	ATTACCO SECONDARIO
Salto sui muri	Salta tra i muri premendo SALTA

In più rispetto al *Marauder*, Il *Marauder Avanzato* può usare un'attacco elettrico.Per usarlo premere l'ATTACCO SECONDARIO quando si è vicini ad un umano o una struttura. Se si combina il salto con l'elettro-shock su due o più obbiettivi, il primo subirà il massimo del danno, il secondo la meta, il trerzo un terzo e così via. per un periodo di un secondo l'attaccande deve stare sull'obbiettivo.

Dragoon

Costo: 3 Stage: 1

Abilità	Controlli
Morde	ATTACCO PRIMARIO
Balzo	premere ATTACCO SECONDARIO e rilasciarlo

Il *Dragoon* attacca mordendo con l'ATTACCO PRIMARIO e con un Balzo con l'ATTACCO SE CONDARIO. Per Balzare, prima premere ATTACCO SECONDARIO per caricare, poi rilasciare per compiere il balzo in avanti e danneggiare qualsiasi cosa si trovi in traiettoria. Mentre si carica è impossibile saltare e ci si muove a velocità ridotta. Mirare in alto per mentre si carica per balzare in alto.

Advanced Dragoon

Costo: 4 Stage: 3

Abilità	Controlli
Morde	ATTACCO PRIMARIO
Balzo	premere ATTACCO SECONDARIO e rilasciarlo
Sparo bavoso	ATTIVA UPGRADE

In più rispetto al *Dragoon* Il *Dragoon Avanzato* può sputare a lungo raggio premendo ATTIVA UPGRADE. Possono essere tenuti nella riserva fino a tre di queste bave dopo di che quste si rigenerano automaticamente col tempo.

Tyrant

Costo: 5 Stage: 3

Abilità	Controlli
Snippo	ATTACCO PRIMARIO
Carica	premere ATTACCO SECONDARIO e rilasciarlo
Aura di guarigione	Stare vicino ai compagni di squadra per aumentare il loro tasso di rigenerazione

Il *Tyrant* attacca snippando con l'ATTACCO PRIMARIO o caricando con l'ATTACCO SECONDARIO. Per caricare prima premere l'ATTACCO SECONDARIO poi muoversi in avanti per correre, quando ri rilascia si corre ad alta velocità per breve tempo, danneggiando qualsiasi cosa sul percorso. Il *Tyrant* ha anche un aura che rigenera il doppio più velocemente gli alieni di minor classe nei paraggi.

2.1.2 Strutture

Tutte le strutture aliene devono essere costruite in prossimità di un uovo (*Egg) o* dell'*Overmind*. Inoltre richiedono l'*Overmind* per funzionare. Tutte le strutture create radicano nel loro intorno rallentando i movimenti dei giocatori umani. Quando distrutte le strutture aliene esplodono, proiettando una pioggia di acido dannoso per gli umani. tutte le strutture possono essere costruite sul pavimento e quando *l'Advanced Grangers* diviene disponibile alcune strutture possono essere costruite su muri e soffitti.

Overmind

Sentience: 0 Stage: 1

L'*Overmind* è la coscienza collettiva che controlla le strutture aliene nella mappa e la capacità degli alieni ad evolversi in classi avanzate. Ci può essere un solo *Overmind* e deve essere completato prima che tutte le altre strutture possano essere sviluppate. Se l'*Overmind* viene distrutto tutte le strutture intorno alle uova smettono di funzionare e gli alieni perdono la capacità di evolversi fino a quando un nuovo *Overmind* viene costruito. L'*Overmind* ha una quantità limitato si 'sentience' (Sentience o Building point) che è distribuito fra ogni altra struttura sviluppata, ciascuno relativo proprio costo.

Egg

Sentience: 10 Stage: 1

L'*Egg (uovo)* è la struttura aliena fondamentalmente e più importante; da qui nascono gli alieni. Sono inoltre l'unica struttura che continua a funzionare in assenza di un *Overmind.* Le uova possono essere costruite sui soffitti.

Acid Tube

Sentience: 8 Stage: 1

Gli *Acid Tubes (Tubi di acido)* sono la struttura difensiva primaria della squadra aliena. Avvicinati a da un essere umano espellono l'acido mortale in tuttle direzioni, anche sopra altre strutture. Gli *Acid Tubes* possono essere costruiti sulle pareti e sui soffitti.

Barricade

Sentience: 10 Stage: 1

Le *Barricades (Barricate)* sono utilizzate per ostruire i corridoi e le porte, ostacolando il movimento ed la linea di tiro umani.

Trapper

Sentience: 8 Stage: 2

I *Trappers* sparano un bolo di adesivo a ogni umano che si trovi nella sua linea di tiro, congelandolo sul posto; se è equipaggiato con la a *Battlesuit*, restringe l'abilità di girare. I *Trappers* possono essere costruiti sulle pareti e sui soffitti, hanno scarso effetto se costruiti sul pavimento.

Booster

Sentience: 12 Stage: 2

Tutti gli alieni che toccano un un *Booster* vengono forniti di un aumento del veleno su tutti gli loro attacchi ravvicinati per un tempo limitato. L'Avvelenamento delle vittime causa una perdita costante di salute con il trascorrere del tempo a meno che gli avversari non usino un Medkit o si ricarichino sulla *Medistation*. Il veleno non funziona contro gli esseri umani dotati di un *Battlesuit*. Il *Booster* inoltre il raddoppia il tasso di rigenerazione di tutti gli alieni vicini con l'eccezione dei *Tyrants*. L'aura di guarigione di un *Tyrant* non è cumulativo con l'effetto di guarigione dei *Booster* .

Hovel

Sentience: 0 Stage: 3

L'*Hovel* è lo scudo corazzato dei *Grangers* che possono nascondersi dentro in caso di necessita. Ci può soltanto essere un *Hovel*. Ci si può entrare e uscire usando il tasto STRUTTURA/EVOLVI (**predefinito Q** N.d.T.)

Hive

Sentience: 12 Stage: 3

L'*Hive (Alveare)* è la casa di milioni di piccoli insettoidi alieni. Quando un essere umano si avvicina alla struttura gli insetti lo attaccano. Gli *Hive* possono essere costruiti sui soffitti.

2.2 Umani

2.2.1 Armi

gli umani nascono (dagli *spawn)* con il *Construction Kit* or il *Rifle*. Non appena guadagnano crediti possono coprare o cambiare armi dall'*Armoury (Armeria)*. Le munizioni possono essere ricaricate per un arma normale all'*Armouries*, o del *Reactors (Reattore)* e dal *Repeaters (ripetitore)* per le armi ad energia, gratuitamente. I giocatori possono portare solo un arma per volta fatta a eccezzione per il *Blaster*. In generale gli esseri umani contano sulle armi della lunga gittata per compensare per la loro mancanza di agilità nei confronti della squadra aliena.

Construction Kit

Costo: 0 Stage: 1

Il *Construction Kit* è il mezzo che gli umani usano per costruire le loro strutture. Il pulsante di ATTACCO PRIMARIO farà comparire un menu con le strutture che è possibile costruire. Dopo aver scelto la struttura, una sagoma colorata comparirà e quando questa sarà verde, premendo il pulsante di ATTACCO PRIMARIO la struttura sarà piazzata. Quando la sagoma è rossa non si potrà costruire. Per rimuovere una struttura in costruzione, premere il pulsante di ATTACCO SECONDARIO; per rimuovere una struttura già piazzata, premere il trasto DECOSTRUISCE STRUTTURA. Dopo aver costruito o decostruito una struttura comparirà un timer nell'angolo in basso a destra dello schermo: fino a quando non sarà finito, non potrai costruire, riparare o decostruire nessuna struttura. Le strutture danneggiate possono essere riparate premendo il pulsante di ATTACCO SECONDARIO vicino ad esse.

Advanced Construction Kit

Costo: 0 Stage: 2

Al secondo stage un kit di costruzione avanzato *(Advanced Construction Kit)* diventerà disponibile per costriure più strutture.

Blaster

Costo: 0 Stage: 1

Il *Blaster* è l'arma secondaria standard degli umani. Ogni giocatore comparirà automaticamente con uno di essi e non potrà essere scambiato con un altra arma. Il *Blaster* spara proiettili deboli e non usa colpi.

Rifle

Costo: 0 Stage: 1

Il *Rifle* è l'arma base degli umani ed è disponibile alla comparsa. Spara velocemente colpi abbastanza accurati ed ha un caricatore da 30 colpi. Fino a 6 caricatori extra possono essere portati per la ricarica.

Pain Saw
PIC
Costo: 100 Stage: 1

Il *Pain Saw* è una potente arma da corpo a corpo che emette scariche elettriche statiche quando usata. Non usa colpi.

Shotgun

Costo: 150 Stage: 1

Lo *Shotgun (fucile a pompa)* spara 8 pallettoni ad ampio raggio ed è l'ideale scontri ravvicinati. Ha un caricatore da 8 colpi e 3 extra.

Las Gun

Costo: 250 Stage: 1

Il *Las Gun (pistola laser)* è simile al rifle ma è più precisa, potente e lenta a sparare e non usa caricatori. E' una arma a energia e perciò deve essere ricaricata al reattore o a un ripetitore. Può tenere fino a 200 celle energetiche, o 300 con un battery pack.

Mass Driver

Costo: 350 Stage: 1

Il *Mass Driver* spara potenti e precisi colpi ma lentamente. E' un arma a energia e ha 5 colpi nel caricatore, o 7 con un battery pack Fino a 4 caricatori extra possono essere tenuti dal giocatore.

Chaingun

Costo: 400 Stage: 1

La *Chaingun* è una potente, imprecisa arma che spara proiettili molto velocemente. Può tenere fino a 300 proiettili ed è meglio usarla da abbassati per ridurre il rinculo dell arma. Gli umani con la battlesuit non avranno il rinculo.

Pulse Rifle

Costo: 400 Stage: 2

Il *Pulse Rifle* è un arma a energia che spara colpi ad alta velocità. Ha 50 celle energetiche per caricatore oppure 75 con un battery pack. Possono essere tenuti 4 caricatori extra.

Grenade

Costo: 200 Stage: 2

La *Grenade (Granata)* è un arma da lancio altamente esplosiva. Viene tirata a una corta distanza premendo il pulsante di attivazione potenziamento
Dopo 4 secondi esploderà causando danni a tutto ciò che si trovava nella area dell'esplosione.

Flamethrower

Costo: 450 Stage: 3

Il *Flamethrower (lanciafiamme)* è un arma incendiaria a corto raggio. Tiene 150 colpi e può facilmente ferire il giocatore che lo usa male

Lucifer Cannon

Costo: 600 Stage: 3

IL *Lucifer Cannon* è l'arma umana più devastante. E' un arma a energia che puo tenere 90 o 135 celle energetiche con un battery pack tenendo premuto il pulsante di attacco primario verrà caricato un lento proiettile con un danno ad ampio raggio. Più tempo si caricherà il colpo e più esso sarà potente e consumerà piu energia. Se caricato troppo a lungo, l'arma esploderà ferendo il giocatore.
L'attacco secondario spara un colpo più piccolo che non richiede caricamento.

Un giocatore umano può equipaggiarsi con i seguenti upgrade, con alcune eccezioni: il *Jet Pack* e il *Battery Pack* non possono essere usati insieme e la *Battlesuit* non può essere usata con il *Jet Pack*, *Battery Pack*, *Light Armour*, o l'E*lmetto*. Si può dunque portare solo un tipo di upgrade per volta. Gli Upgrade che non hanno un'effetto intrinseco possono essere selezionati dal giocatore con il pulsante PROSSIMO UPGRADE e PRECEDENTE UPGRADE e attivati con ATTIVA UPGRADE (**predefinito Rotellina Mouse** N.d.T.).

Light Armour

Costo: 70 Stage: 1

Light Armour (Armatura Leggera) garantisce indossandolo una miglior difesa sul torso e sulle gambe.

Helmet

Costo: 90 Stage: 2

Helmet (Elmetto) migliora la difesa sulla testa ed inoltre visualizza un radar che mostra le posizioni relative ai dei nemici vicini e delle strutture nemiche.

Medkit
Costo: 0 Stage: 1

Il *Medkits* è subito disponibile quando un umano "nasce" ed è automaticamente ricaricato tramite la *Medistations* una volta raggiunta la piena salute. Non può essere venduto o comprato all'*Armoury*. Quando utilizzato tramite il tasto ATTIVA UPGRADE, Il *Medkits* restituisce la salute dapprima lentamente poi sempre a maggior velocità, qualsiasi danno ricevuto durante il suo utilizzo verrà velocemente guarito. Inoltre gli umani avvelenati possono usare il *Medkit* per curare l'avvelenamento e diventare immuni per 30 secondi.

Battery Pack

Costo: 100 Stage: 1

Il *Battery Pack* aumenta la capacità massima di stoccaggio delle armi ad energia del 50%. Non può essere usato con il *Jet Pack*.

Jet Pack

Costo: 120 Stage: 2

Il *Jet Pack* garantisce a chi lo indossa la capacità di volare in manera illimitata. Si attiva tramite ATTIVA UPGRADE , il giocatore può salire e scendere premendo SALTA e ABBASSATI. The *Jet Pack* smette di funzionare quando il *Reactor* non è operativo. Inoltre è soggetto a malfunzionamento se il giocatore viene ferito. Il *Jet Pack* non può essere usato con il *Battery Pack*.

Battlesuit

Costo: 400 Stage: 3

La *Battlesuit* provvede ad aumentare in maniera significativa le difese su tutto il corpo. La *Battlesuit* non può essere usata in combiazione con altre protezioni (*Light Armour*,

Helmet, *Battery Pack*, and *Jet Pack*). I giocatori non possono accucciarsi mentre indossano la *Battlesuit.*

2.2.3 Strutture

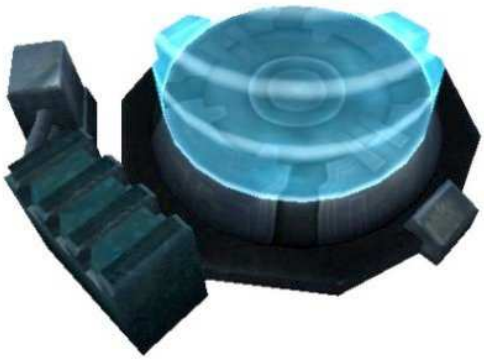
Tutte le strutture umane devono essere costruite in prossimità del *Reactor (Reattore)* o del *Repeater (Ripetitore)*. Con l'unica eccezione dei *Telenodes (Telenodi)*, tutte le strutture richiedono la presenza di un *Reactor* attivo per funzionare. Tutte le strutture se distrutte esplodono con tale potenza da distruggere tutto ciò si trovi nelle immediate vicinanze.

Reactor

Power: 0 Stage: 1

Il *Reactor (Reattore)* è la sorgente di alimentazione di tutte le strutture umane nella base. Ci può essere un solo *Reactor*, e deve essere presente prima che tutte le strutture tranne i *Repeaters (ripetitori)* possano essere sviluppate. Se il *Reactor* viene distrutto tutte le strutture circostanti smettono di funzionare tranne i *Telenodes*. il *Reactor* ha un "Voltaggio" limitato (Power o Building point) la distrbuzione dell'energia è suddivisa tra tutte le strutture sviluppate, ciascuna relativa al proprio costo.

Telenode



Power: 10 Stage: 1

I *Telenode (Telenodi)* sono la più basilare e fondamentale struttura umana; da qui gli umani entrano in gioco. sono anche le uniche strutture che continuano a funzionare con l'assenza del *Reactor*.

Machine Gun Turret

Power: 8 Stage: 1

La *Machine Gun Turret (Torretta)* e la primaria struttura di difesa degli umani. Quando hanno una linea di tiro sgombra attaccano gli alieni che si trovino nel loro raggio d'azione con una scarica di colpi e seguono l'alieno fino a che questo non sia morto (o fuori dal raggio d'azione N.d.T.).

Tesla Generator

Power: 10 Stage: 3

I *Tesla Generators (Generatori Tesla)* Sono una struttura di difesa che colpisce incondizionatamente tutti i bersagli che si trovino sotto il loro raggio d'azione con una scarica elettrica. per essere costruiti e funzionare i *Tesla Generator* richedono la preseza del *Defense Computer* in qualche posto della mappa.

Armoury

Power: 10 Stage: 1

L'*Armoury (Armeria)* e una parte essenziale della base umana, permettendo che gli aggiornamenti oltre l'apparecchiatura di disposizione degli spawn *(Construction Kit)* siano comprati e scambiati. E' il solo modo che un umano ha per "Evolversi"(tecnologicamente). Per utilizzare un *Armoury*, avvicinarsi e premere il tasto Usa Structtura/Evolvi. Le munizioni per le armi non-energetiche possono essere aquistate solamente qui gratuitamente utilizzando il tasto Acquista Munizioni (**predefinito B** N.d.T.).

Defense Computer

Power: 8 Stage: 2

Il *Defense Computer (Computer di Difesa)* coordina gli attacchi delle *Machine Gun Turrets*, evitando che esse sparino ad un solo bersaglio quando ce ne sono altri disponibili. E' richiesto per la produzione dei *Tesla Generator*.

Medistation

Power: 8 Stage: 1

La *Medistation (Stazione Medica)* fornisce gli unici mezzi agli esseri umani per guarirsi. Restando in piedi su di essa, un essere umano rigenererà rapidamente la salute fino al massimo di 100. Di provides the only means for humans to heal themselves. By standing on one, a human will quickly regenerate health up to their maximum of 100. La *Medistation* ricarica anche i *Medkits* degli umani una volta raggiunta la piena salute. Solo un persona per volta può usare la *Medistation*.

Repeater

Power: 0 Stage: 2

I *Repeaters (Ripetitori)* servono da distributori di alimentazione che possono essere costruiti dovunque non ci sia alimentazione, anche quando non c'è nessun *Reactor* presente. Qualunque altra struttura può essere sviluppata nella prossimità ad un ripetitore funzionante come se fosse un *Reactor*. Se un *Repeater* non alimenta niente per 90 secondi, viene automaticamente distrutto.

3 Documentazione Tecnica

3.1 Bindings

Name in menu	Binding	Funzione
ATTACCO PRIMARIO	**+attack**	Usa ATTACCO PRIMARIO.
ATTACCO SECONDARIO	**+button5**	Usa ATTACCO SECONDARIO.
PRECEDENTE UPGRADE	**weapprev**	Se umano seleziona il precedente upgrade dall'inventario.
PROSSIMO UPGRADE	**weapnext**	Se umano seleziona il prossimo upgrade dall'inventario.
ATTIVA UPGRADE	**+button2**	Se umano attiva l'upgrade selezionato dall'inventario. (solo per alcune abilità)
RICARICA	**reload**	Se umano ricarica l'arma selezionata.
COMPRA MUNIZIONI	**buy ammo**	se umano compra munizioni da un *armoury*, *repeater* o *reactor*.
USA MEDIKIT	**itemact medkit**	Se umano aziona il *Medkit*.
USA STRUTTURA/EVOLVI	**+button7**	Se umano usa la struttura di fronte al giocatore. Se alieno evolve in classi differenti.
DECOSTRUISCE STRUTTURA	**deconstruct**	Se costruttore decostruisce la struttura di fronte al giocatore.
SPRINT	**boost**	Corre veloce.
–	**destroy**	Se costruttore, distrugge la struttura di fronte al giocatore.
–	**itemact <item>**	Se premuto attiva uno specifico item. Seleziona le armi.
–	**itemdeact <item>**	Se premuto deattiva uno specifico item.
–	**itemtoggle <item>**	If held, toggle the state of the specified item.
–	**sell <item>**	If held and within range of an *armoury*, sell the specified item.
–	**sell weapons**	Se premuto nelle vicinanze della *armoury*, vende tutte le armi.
–	**sell upgrades**	Se premuto nelle vicinanze della *armoury*, vende tutte gli *upgrade*.
–	**buy <item>**	Se premuto nelle vicinanze della *armoury*, compra l'item specificato.
–	**class <class>**	Se si hanno sufficenti kill, evolve nella classe specificata.
–	**Costruisce<structure>**	Se costruttore costruisce la struttura specificata.

<item> – *blaster, rifle, ckit, ackit, shotgun, lgun, prifle, mdriver, flamer, chaingun, lcannon, psaw, gren, medkit, jetpack*

<class> – *builder, builderupg, level0, level1, level1upg, level2, level2upg, level3, level3upg, level4*

<structure> – *eggpod, barricade, booster, acid_tube, hive, trapper, overmind, hovel, telenode, medistat, mgturret, tesla, dcc, arm, reactor, repeater*

3.2 Sistemi particellari

File abbinati a pattern scripts/*.particle sono caricate come file di descrizione di un sistema della particellare. Ogni file particellare può contenere un numero arbitrario di sistemi particellari, tanti quanti un file shader può ospitare altri shaders. Un sistema particellare è dichiarato da un nome seguito da "curly braces" la quale la funzionalità del sistema particellare è definito. Per esempio:

```
aShinyNewParticleSystem { }
```

Inside the particle system declaration are placed up to four particle ejectors. Ejectors are identified by the keyword ejector and curly braces:

```
aShinyNewParticleSystem
{
  ejector { }
```

```
  ejector { }
  thirdPersonOnly
}
```

The thirdPersonOnly keyword may be used to specify that the particle system is not visible from the first person if it relates to that client. The role of the particle ejector is to create some number of new particles at a defined rate. These attributes are Controlliled by the following parameters:

- *count <number>|infinite* - the number of particles this ejector will spawn.
- *delay <msec>* - the delay in msec before the ejector starts spawning.
- *period <initial> <final> <variance>* - the period between particle ejections.

It is perfectly acceptable to have an initial period of zero. In this case the number of particles specified by the count keyword will be ejected at once. It is not permissible to have count infinite and a period of zero for obvious reasons.

At ejection time each ejector creates up to four new particles based on templates. These are specified in the ejector section using the particle keyword:

```
aShinyNewParticleSystem
{
  ejector
  {
    particle { }
    particle { }
    count 50
    delay 0
    period 0 - 0
  }
}
```

Alcuni particle template contengono questi attributi:

- *shader <fps>|sync <shader1> <shader2> ... <shaderN>* - this specifies the shaders to use for the particle. The frame rate can be set to a static rate or the *sync* parameter can be used in which case the frame rate will be synchronised to the lifetime of the particle such that the first frame is displayed on birth and the last frame is displayed immediately before death.
- model <model1> <model2> ... <modelN> - use one of the specified models as the particle. This cannot be used in conjunction with the shader keyword.
- modelAnmation <firstFrame> <numFrames> <loopFrames> <fps>|sync - animation parameters to use when model particles are employed.
- *displacement <x> <y> <z> <variance>* - a static displacement about the attachment point. The *variance* parameter specifies a random displacement in all axes.
- *normalDisplacement <displacement>* - for particle systems that have their normal set (impact particle systems for example) this specifies the magnitude of a displacement along the normal.
- *velocityType static|static_transform|tag|cent|normal* - this specifies how the particle will compute its initial velocity. *static* means it is specified statically in the .particle file, *static_transform means the same, except that it is transformed by the orientation matrix of what it is attached to, tag* means the velocity is in the direction of the tag it is attached to, *cent* means the velocity is in the direction of the cent it is attached to and *normal* means the velocity is in the direction of the particle system normal.
- *velocityDir linear|point* - this specifies whether the initial velocity is computed as a simple direction or as the direction towards a secondary point (defined by *velocityPoint* or dynamically through *velocityType cent*).
- *velocity <x> <y> <z> <variance>* - for when *velocityType static* is present this specifies the direction. The *variance* here is specified in degrees e.g. "~5" - up to 5 degrees deviation.
- *velocityMagnitude <magnitude>* - the magnitude of the velocity.
- *velocityPoint <x> <y> <z> <variance>* - for when *velocityType static* and *velocityDir point* are present this specifies the point to move towards.
- *parentVelocityFraction <fraction>* - for when the particle system is attached to a cent this specifies the fraction of the cent's velocity that is added to the particle's velocity.
- *accelerationType static|static_transform|tag|cent|normal* - this specifies how the particle will compute its acceleration. *static* means it is specified statically in the .particle file, *static_transform means the same, except that it is transformed by the orientation matrix of what it is attached to, tag* means the acceleration is in the direction of the tag it is attached to, *cent* means the acceleration is in the direction of the cent it is attached to and *normal* means the acceleration is in the direction of the particle system normal.
- *accelerationDir linear|point* - this specifies whether the acceleration is computed as a simple direction or as the direction towards a secondary point (defined by *accelerationPoint* or dynamically through *accelerationType cent*).
- *acceleration <x> <y> <z> <variance>* - for when *accelerationType* static is present this specifies the direction. The *variance* here is specified in degrees e.g. "~5" - up to 5 degrees deviation.
- *accelerationMagnitude <magnitude>* - the magnitude of the acceleration.
- *accelerationPoint <x> <y> <z> <variance>* - for when *accelerationType static* and *accelerationDir point* are present this specifies the point to move towards.
- *bounce <fraction>|cull* - the fraction of velocity that is reflected when a particle collides. If this is set to 0.0 the particle won't collide. When *cull* is used particles are culled as soon as they collide with objects.
- bounceMark <count> <radius> <shader> - make a mark at each bounce point for up to <count> bounces.
- bounceSound <count> <sound> - make a sound at each bounce point for up to <count> bounces.
- dynamicLight <delayRadius> <initialRadius> <finalRadius> { <r> <g> <b> } - attach a dynamic light to this particle.
- color <delay> { <ir> <ig> <ib> } { <fr> <fg> <fb> } - color the particle where <i.> refers to the initial color component and <f.> refers to the final color component.
- *overdrawProtection* - cull particles that occupy a large amount of screen space.
- *realLight* - light particles using the lightgrid instead of fullbright.
- *cullOnStartSolid* - cull particles that are spawned inside brushes.
- *radius <delay> <initial> <final>* - the radius of the particle throughout its lifetime. The *delay* parameter specifies the time in msec before radius scaling begins. The *initial* and *final* parameters specify the radii of the particle in quake units.
- *alpha <delay> <initial> <final>* - the alpha of the particle throughout its lifetime. The *delay* parameter specifies the time in msec before alpha scaling begins. The *initial* and *final* parameters specify the alpha of the particle where 1.0 is totally opaque and 0.0 is totally transparent.
- *rotation <delay> <initial> <final>* - the rotation of the particle throughout its lifetime. The *delay* parameter specifies the time in msec before the rotation begins. The *initial* and *final* parameters specify the rotation of the particle in degrees.
- *lifeTime <time>* - the lifetime of the particle.
- *childSystem <particle system>* - specifies a particle system to attach to this particle.
- childTrailSystem <trail system> - specifies a trail system to attach to this particle.
- *onDeathSystem <particle system>* - specifies a particle system to spawn at the point where this particle died.

Except for vector components, *shader fps ...* and *period <initial <final> <variance>*, every value can be specified with a random variance. The syntax for this is as follows:

```
[value][~variance[%]]
```

So the following forms are possible, where random is a random number between 0.0 and 1.0 inclusive:

```
5.0     // 5.0
5.0~8.0  // 5.0 + ( random * 8.0 )
5.0~200% // 5.0 + ( random * 5.0 * 200% )
~7.0     // random * 7.0
```

This allows for relatively flexible randomisation of most of the particle's parameters. For parameters taking an initial and final value, specifying the final value as '-' will result in a final value the same as the initial value.

For the purposes of map based particle systems using *misc_particle_system* it is safe to ignore *velocityType* and *accelerationType tag|cent|normal*, *normalDisplacement* and *parentVelocityFraction* altogether.

Of course, it is not necessary to specify every parameter documented here for every particle system. If a parameter is not included it will usually default to zero. C/C++ style comments can be used throughout. There are an enormous number of possible combinations of particle systems parameters and as such it is impractical to test them all. For this reason it is possible that certain permutations do not behave as expected or wrongly. In this case you may have discovered a bug - let us know. Having said this when you're having problems with a particle system make sure you scroll up the console and check that it compiled OK, I've written the parser to be very intolerant of error.

Here is an example particle system:

```
aShinyNewParticleSystem
{
```

```
 ejector
 {
  particle
  {
   shader sync shader1 shader2


   velocityType static
   velocityDir linear
   velocityMagnitude 200
   velocity 0 0 1 ~30
   accelerationType static
   accelerationDir linear
   accelerationMagnitude 50
   acceleration 0 0 1 ~0
   radius 0 10.0 50.0
   alpha 0 1.0 1.0
   rotation 0 ~360 -
   bounce 0.4
   lifeTime 1500
  }
  count 50
  delay 0
  period 0 - 0
 }
}
```

3.3 Trail System

Files matching the pattern scripts/*.trail are loaded as trail system description files. Each .trail file can contain an arbitrary number of discrete trail systems, much like a .shader file can house many shaders. A trail system is declared by a name followed by curly braces within which the functionality of the trail system is defined. For example:

```
aShinyNewTrailSystem { }
```

Inside the particle system declaration are placed up to four trail beams. Beams are identified by the keyword beam and curly braces:

```
aShinyNewTrailSystem
{
 beam { }
 beam { }
 thirdPersonOnly
}
```

The thirdPersonOnly keyword may be used to specify that the trail system is not visible from the first person if it relates to that client. A trail beam describes the appearance of one element of the trail system:

- *shader <shader> - the shader to use to texture this beam.*
- segments <number> - the number of quads that make up the beam.
- width <frontWidth> <backWidth> - the width of the beam at the front and back.
- alpha <frontAlpha> <backAlpha> - the alpha of the beam at the front and back.
- color { <fr> <fg> <fb> } {
 <bg> <bb> } - the color of the beam at the front and back.
- segmentTime <time> - how long a single segment lasts when the trail is only attached at one end.
- fadeOutTime <time> - how long this beam takes to fade away.
- textureType [stretch <frontTC> <backTC>]|[repeat [front|back] <repeatLength>] - how to texture the beam. stretch causes the texture to be stretched from the front to the back using the specified texture coordinates. repeat causes the texture to be repeated over a specified length either from the front or the back.
- model <model1> <model2> ... <modelN> - use one of the specified models as the particle. This cannot be used in conjunction with the shader keyword.
- modelAnmation <firstFrame> <numFrames> <loopFrames> <fps>|sync - animation parameters to use when model particles are employed.
- *realLight* - light particles using the lightgrid instead of fullbright.
- jitter <magnitude> <period> - this specifies a random jitter of the position of each beam node by magnitude every period.
- jitterAttachments - if this is specified the end points of the beam are jittered as well as the intervening nodes.

3.4 Map Rotation System

The file *maprotation.cfg* is used to describe up to 16 map rotations which may be used by a Tremulous server. In its most simple form, the syntax is as follows:

```
mapRotation1
{
 map1
 map2
 map3
}
mapRotation2
{
 map6
 map3
 map9
}
```

This specifies two rotations, each consisting of three maps. The contents of the cvar *g_initialMapRotation* specifies the map rotation to start after the map the server was started with has finished. It is possible to specify a list of server commands to be run after a map has finished:

```
mapRotation3
```

```
{
  map1
  {
    set sv_hostname ıJust finished map1!j
    set g_teamForceBalance 0
  }

  map2
  {
    set g_teamForceBalance 1
  }
  map3
}
```

Primitive logic is also available:

```
mapRotation4
{
  map1
  goto map3
  map2
  if numClients > 8
    mapRotation3
  map3
  if lastWin aliens
    mapRotation2

  if random
    mapRotation1
}
mapRotation5
{
  map1
  if lastWin humans
    map4
  map2
  map3
  goto map1
  map4
  map5
}
```

The **goto** keyword is used to unconditionally branch to either another map *in the current rotation* or another map rotation entirely. The **if** keyword is used in conjunction with a condition to decide whether or not to branch to the specified map or rotation (as with the **goto** keyword). The condition itself can be one of **numClients <op> <number>**, **lastWin <team>** or **random**, where **<op>** is **<**, **>** or **=** and **<team>** is **aliens** or **humans**. The **random** condition simply chooses whether or not to execute the change randomly, with each outcome equally likely.

4 Credits

Tim *'Timbo'* Angus – Programming and Direction

Nick *'jex'* Jansens – Mapping, texturing and 2D artwork

Robin *'OverFlow'* Marshall – Modelling, animation and mapping

Jan *'Stannum'* van der Weg – Texturing and mapping

Mike *'Veda'* Mcinnerney – Modelling, animation and texturing

Gordon *'Godmil'* Miller – Mapping

'Who-[Soup]' – Mapping

Tristan *'jhrx'* Blease – Mapping

Paul *'MoP'* Greveson – Modelling and texturing

Chris *'Dolby'* McCarthy – Sound

Special thanks

Asa *'Norfenstein'* Kravets – Manual content, QA, design and balance suggestions

'Crylar' – Concept art

Yves *'evillair'* Allaire – Textures

Randy *'ydnar'* Reddig – Textures

Richard *'R1CH'* Stanway – Server hosting

Stéphane *'MEGASTeP'* Peter – Early test server hosting

Sourceforge and TARDIS – Web hosting

icculus.org – Subversion hosting

The contributors to icculus.org/quake3/ – Various

Arsonide, Bajoran, Bt, Chamooze, Crylar, Cybernetsam, dzjepp, ectox, Edo, evil poop, Excalibur, FroggyQuim, Idle Wild, juice, Kai, kingping, Lava Croft, MajorPain, MiDiaN, Molog, Mutemode, Norfenstein, Orc, R1CH, Ratti, Ravyn, Salteh, Sandy, SharkDog, slux, Suddien, Supa, Survivor, Swie, sysrq, TerrorEast, Tyler, Vitae, Woo – Beta testers

Also thanks

babyomen, Carc, djbob, Grim, Grytviken, Gumby, heimdall, Hellbringer, Hentai, Mighty_Pea, Psylo, Reaper-1, RR2D02, Saig, Smack, T-bone, The GtkRadiant people, The inhabitants of Quake3World, ThePyro, TTimo, ValouR

Tradotto in Italiano da:

Chester 885[ITA]

[DH]Merope[ITA]