

3D ΜΟΝΤΕΛΟΠΟΙΗΣΗ

# Blender: Επεκτάσεις με την Python



**ΜΕΡΟΣ 5ο** Σε αυτό το τεύχος θα μάθουμε πώς να επεκτείνουμε τη λειτουργικότητα του Blender, χρησιμοποιώντας την Python...



## ΤΟΥ ΧΡΗΣΤΟΥ ΣΜΑΪΛΗ

Ο Χρήστος έκανε τα πρώτα βήματά του στο Blender πριν από τέσσερα χρόνια, αλλά ασχολείται και με τον προγραμματισμό. Μπορείτε να επικοινωνήσετε μαζί του στο [greekblend@yahoo.gr](mailto:greekblend@yahoo.gr)



Τις προηγούμενες φορές είδαμε πώς μπορούμε να δημιουργήσουμε μοντέλα και animation στο Blender. Σε αυτό το τεύχος, όμως, θα ασχοληθούμε με κάτι εντελώς διαφορετικό. Θα μάθουμε να φτιάχνουμε scripts που θα επεκτείνουν το Blender, χρησιμοποιώντας τη γλώσσα Python. Σε αντίθεση με άλλα προγράμματα, το Blender δεν είναι στατικό. Δηλαδή, μπορείτε να βελτιώσετε τη λειτουργικότητά του χωρίς να χρειάζεται να αλλάξετε τον πηγαίο κώδικα και να τον μεταγλωττίσετε. Υπάρχουν 2 τρόποι για να επεκτείνετε το Blender, το Python Scripting και τα Binary plug-ins. Επειδή, όμως, τα Python Scripts είναι ο πιο διαδεδομένος τρόπος, θα ασχοληθούμε μόνο με αυτή τη μέθοδο. Θα αναλύσουμε, λοιπόν, κάποια από τα βασικά εργαλεία που προσφέρει το Blender μέσω του Python API του, ώστε να φτιάξουμε ένα απλό script, το οποίο θα δημιουργεί αυτόματα terrain με λόφους. Μολονότι ο χρήστης θα μπορεί να δίνει κάποιες τιμές για να καθορίσει το πώς περίπου θα δείχνει το τελικό αποτέλεσμα, κάθε

φορά, το terrain που θα προκύπτει, θα είναι ελαφρώς διαφορετικό από τη προηγούμενη, ακόμη και με τις ίδιες ρυθμίσεις. Πριν προχωρήσουμε, όμως, πρέπει να τονίσουμε πως καλό θα ήταν έχετε ήδη κάποιες βασικές γνώσεις στην Python.

## Ξεκινώντας

Κατ' αρχάς, θα ήταν καλό να τρέξετε το Blender μέσω ενός τερματικού, προκειμένου να βλέπετε όλα τα μηνύματα που σχετίζονται με το debugging του script. Το τερματικό αυτό θα σας φανεί επίσης χρήσιμο στις περιπτώσεις που θέλετε να εισάγετε ή να εξάγετε δεδομένα μέσω της κονσόλας, χρησιμοποιώντας το script (π.χ., με τις εντολές print και raw\_input). Σε αυτό το σημείο, πρέπει να ειπωθεί πως το Blender περιλαμβάνει έναν Python interpreter, αλλά και έναν αρκετά καλό κειμενογράφο, μέσω του οποίου μπορούμε να φτιάξουμε και να τεστάρουμε το script μας. Αφού, λοιπόν, ανοίξετε το Blender, μεταβείτε στον κειμενογράφο, κλικάροντας το κουμπί Window Type

(θα το βρείτε στην αριστερή άκρη του header του Buttons Window) και επιλέγοντας το "Text Editor" από το αναδυόμενο μενού. Μετά, από το μενού File του κειμενογράφου, κάντε κλικ στο new για να ξεκινήσετε ένα νέο script. Οποιαδήποτε στιγμή θελήσετε να το εκτελέσετε, μπορείτε να πατήσετε τα πλήκτρα ALT+P. Εναλλακτικά, μπορείτε να πατήσετε την εγγραφή Run Python Script από το μενού File. Αν υπάρχει κάποιο σφάλμα στο κώδικα, τότε, όταν εκτελέσετε το script, θα εμφανιστεί σχετικό μήνυμα, το οποίο θα σας προτρέπει να κοιτάξετε το τερματικό από το οποίο τρέξατε το Blender, προκειμένου να ενημερωθείτε για την αιτία του σφάλματος.

Ας αρχίσουμε, όμως, να γράφουμε κώδικα! Για αρχή, θα εισάγουμε τα modules του Blender API της Python που θα χρειαστούμε:

```
import Blender
from Blender import Draw,Mesh,Object,Scene,Modifier,Mathutils
```

Το "Blender" της 1ης γραμμής είναι το βασικό module του API. Μέσα από αυτό, αποκτούμε πρόσβαση σε submodules, που το καθένα προσφέρει εξειδικευμένη λειτουργικότητα, ανάλογη του ονόματός του (για παράδειγμα, το Mesh module περιέχει λειτουργίες που αφορούν στα μοντέλα, το Scene αφορά στα περιεχόμενα της σκηνής 3D κ.ο.κ.). Ας συνεχίσουμε, όμως, για να δούμε πώς μπορούμε να προσθέσουμε ένα plane στη σκηνή μας, το οποίο θα αποτελέσει τη βάση πάνω στην οποία θα φτιάξουμε το terrain. Για να το κάνετε αυτό, γράψτε τον ακόλουθο κώδικα (υπάρχει στο συνοδευτικό DVD):

```
def
createterrain(size,cliff1y,cliff2y,cliff3y,cliff4y,randomness):
#Δημιουργούμε μία "Mesh" οντότητα με όνομα "mountain"
object=Object.New("Mesh","mountain")
#Φτιάχνουμε ένα Plane με μέγεθος size(=5)
mountain=Mesh.Primitives.Plane(size)
#Συνδέουμε το μοντέλο με την οντότητα object
object.link(mountain)
#Αντιστοιχίζουμε τα δεδομένα της σκηνής στη scene
scene=Scene.GetCurrent()
#συνδέουμε το object με τη σκηνή
scene.link(object)
#Εμφανίζουμε το plane
Blender.Redraw()
```

Όπως βλέπετε, δημιουργούμε μία νέα συνάρτηση με όνομα createterrain, η οποία θα είναι υπεύθυνη για τη δημιουργία του terrain μας. Στην πρώτη εντολή της, χρησιμοποιούμε τη συνάρτηση Object.New που ανήκει στο submodule Object για να δημιουργήσουμε μία νέα 3D οντότητα. Η συνάρτηση αυτή δέχεται τις εξής παραμέτρους: 1) τον τύπο της οντότητας, που στη δική μας περίπτωση είναι ένα μοντέλο ("Mesh"), και 2) το όνομα της οντότητας (mountain). Σε αυτό το σημείο να διευκρινίσουμε πως το submodule Object μάς προσφέρει συναρτήσεις και πληροφορίες που αφορούν σε

## ΚΑΘΟΡΙΖΟΝΤΑΣ ΤΑ PATHS ΤΗΣ PYTHON...

Οές οι συναρτήσεις που χρησιμοποιούμε σε αυτό το παράδειγμα, περιλαμβάνονται στο Blender API. Όμως, στα δικά σας σκριπτάκια ίσως χρειαστεί να χρησιμοποιήσετε και άλλα modules που προσφέρει η Python (π.χ., το module math). Σε αυτή τη περίπτωση, πρέπει να εγκαταστήσετε την Python και να δώσετε στο σύστημα τα paths που αντιστοιχούν στην εγκατάστασή της. Αν έχετε εγκαταστήσει την Python από το package manager της διανομής σας, τότε το path της θα έχει δοθεί ήδη στο σύστημα.

Όμως, αν μεταγλωττίσατε την Python, θα πρέπει να κάνετε αυτή τη διαδικασία χειροκίνητα. Σε αυτήν την περίπτωση, ακολουθήστε την ακόλουθη διαδικασία. Ανοίξτε ένα τερματικό και δώστε την εντολή python. Για να βρείτε τα paths της γλώσσας, δώστε import sys και print sys.path. Η έξοδος που θα λάβετε, θα μοιάζει με την ακόλουθη (σε εσάς θα διαφέρει λίγο):

```
['/ramdisk/home/dsl',
'/usr/local/lib/python25.zip',
'/usr/local/lib/python2.5',
'/usr/local/lib/python2.5/plat-linux2',
'/usr/local/lib/python2.5/lib-tk',
'/usr/local/lib/python2.5/lib-dynload',
'/usr/local/lib/python2.5/site-
```

packages']

Αυτό που μένει να κάνετε, είναι να ανοίξετε το αρχείο .bashrc, το οποίο θα βρείτε στον προσωπικό φάκελό σας, και να γράψετε εκεί τα paths της προηγούμενης εντολής. Οι γραμμές που θα πρέπει να συμπληρώσετε, θα μοιάζουν κάπως έτσι:

```
export
PYTHONPATH=/usr/local/lib/python25.
zip:/usr/local/lib/python2.5:
/usr/local/lib/python2.5/plat-
linux2:/usr/local/lib/python2.5/lib-
tk:/usr/local/lib/python2.5/lib-
dynload:/usr/local/lib/python2.5/site-
packages
```

Παρατηρήστε πως τα paths που βάλουμε στο .bashrc, είναι αυτά που μας επιστράφηκαν μετά την εκτέλεση της εντολής print sys.path, αλλά χωρίζονται μεταξύ τους με το χαρακτήρα ( : ) και όχι με αποστρόφους ή κόμματα! Αφού σώσετε το αρχείο, αποσυνδεθείτε και επανασυνδεθείτε για να εφαρμοστούν οι αλλαγές. Τώρα, θα πρέπει να μπορείτε να χρησιμοποιήσετε όλα τα modules της Python στο script του Blender που φτιάχνετε!!!

κάθε είδους "3D οντότητα", η οποία μπορεί να υπάρξει σε μία σκηνή του Blender, όπως κάμερες, μοντέλα, φώτα κ.λπ.

Στο επόμενο βήμα, δημιουργούμε το μοντέλο ενός plane. Αυτό το επιτυγχάνουμε, χρησιμοποιώντας 2 submodules. Πρώτα, χρησιμοποιούμε το Mesh submodule (το οποίο μας δίνει πρόσβαση σε λειτουργίες σχετικές τόσο με τη διαχείριση της γεωμετρίας μοντέλων όσο και με τη δημιουργία τους) και μέσα από αυτό καλούμε το submodule Primitives, που περιέχει συναρτήσεις σχετικές με τη δημιουργία των primitive μοντέλων που περιέχει το Blender. Μία από αυτές τις συναρτήσεις είναι και η Plane, που χρησιμοποιούμε για να δημιουργήσουμε το εν λόγω μοντέλο. Η συγκεκριμένη συνάρτηση παίρνει ως μοναδικό όρισμα το μέγεθος του μοντέλου. Το αντικείμενο που επιστρέφεται, αντιστοιχίζεται στη μεταβλητή ονόματι mountain. Με τις εντολές που ακολουθούν, αντιστοιχίζουμε στην οντότητα object το μοντέλο mountain και στη συνέχεια την υπάρχουσα 3D σκηνή του Blender στη μεταβλητή scene, χρησιμοποιώντας τη συνάρτηση GetCurrent() η οποία ανήκει στο submodule Scene. Έπειτα, συνδέουμε την οντότητα object με τη 3D σκηνή και εμφανίζουμε το νέο μοντέλο με τη συνάρτηση Blender.Redraw().

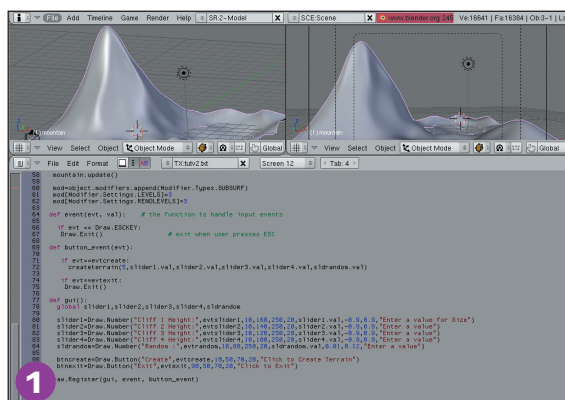
## Προσθέτοντας λόφους...

Στο τεύχος 17 είχαμε δει πώς μπορούμε να φτιάξουμε ένα terrain. Αν θυμάστε, είχαμε διαιρέσει την επιφάνεια του Plane σε περισσότερα τμήματα με το subdivize, ώστε να προσθέσουμε vertices που θα μας βοηθούσαν να σχηματίσουμε λόφους. Επιπρόσθετα, είχαμε κάνει το terrain να φαίνεται βραχώδες, χρησιμοποιώντας το fractal subdivize. Αυτή τη φορά, όμως, θα κάνουμε όλη τη παραπάνω διαδικασία μέσω κώδικα. Συμπληρώστε, λοιπόν, τον ακόλουθο κώδικα στη συνάρτηση createmountain:

```
#κάνουμε 2 φορές subdivize στο plane
mountain.subdivide(0)
mountain.subdivide(1)
#δημιουργούμε τους λόφους
mountain.verts[21].co[2] += cliff1y*size
mountain.verts[22].co[2] += cliff2y*size
mountain.verts[23].co[2] += cliff3y*size
```

## ΓΡΗΓΟΡΗ ΣΥΜΒΟΥΛΗ

Ο κειμενογράφος του Blender υποστηρίζει συντακτικό χρωματισμό και αριθμηση γραμμών. Για να ενεργοποιήσετε το πρώτο, κάντε κλικ στο κουμπί με όνομα AB στο header του κειμενογράφου. Αν θέλετε να ενεργοποιήσετε και την αριθμηση γραμμών, τότε κλικάρετε το κουμπί με τις τρεις γραμμές δίπλα από το κουμπί AB.



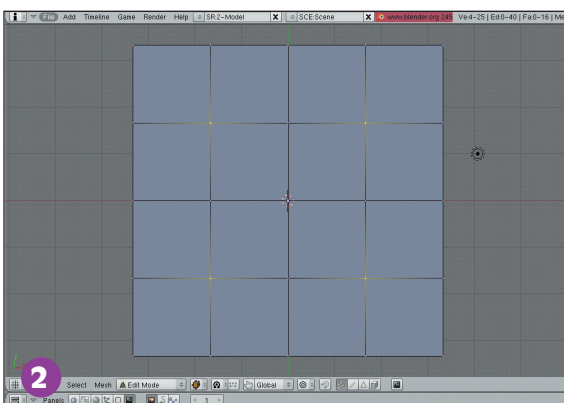
Εδώ μπορείτε να δείτε τον κειμενογράφο του Blender με ενεργοποιημένο το συντακτικό χρωματισμό και την αριθμηση γραμμών.

## ΠΕΡΙΣΣΟΤΕΡΑ ΓΙΑ ΤΟ BLENDER API

Αν θέσετε να μάθετε περισσότερα για το Blender Python API, μπορείτε να συμβουλευτείτε το "The Blender Python API Reference", που θα βρείτε στην ακόλουθη διεύθυνση: [wiki.blender.org](http://wiki.blender.org). Εκεί, στο wiki του Blender, πέρα από πληροφορίες για το Blender API, μπορείτε να βρείτε οτιδήποτε θελήσετε για οποιαδήποτε άλλη λειτουργία του Blender. Ένα ακόμη πολύ καλό μέρος για να δημοσιεύσετε τα scripts σας, να ζητήσετε συμβουλές και απαντήσεις σε απορίες σας ή να δείτε τα scripts που έχουν φτιάξει άλλοι, είναι το τμήμα "Python & Plugins" του φόρουμ του [blenderartists.org](http://blenderartists.org). Τέλος, αν έχετε όρεξη για ακόμη περισσότερα tutorials για τη χρήση του Blender API (και γενικότερα για το Blender), επισκεφθείτε το επίσημο site του Blender ([www.blender3d.org](http://www.blender3d.org))

```
mountain.verts[24].co[2] += cliff4y * size
#εξομαλύνουμε τις πλευρές του μοντέλου
for i in mountain.faces:
    i.smooth=1
# Ξανά subdivίde στο μοντέλο
mountain.subdivide(1)
mountain.subdivide(1)
# κάνουμε την επιφάνειά του πιο βραχώδη
for i in mountain.verts:
    i.co[0] += Mathutils.Rand(0.0,randomness) * size
    i.co[1] += Mathutils.Rand(0.0,randomness) * size
    i.co[2] += Mathutils.Rand(0.0,randomness) * size
#εμφανίζουμε τις αλλαγές στο 3D view
mountain.update()
```

Αρχικά καλούμε δύο φορές τη συνάρτηση subdivίde, η οποία ανήκει στο submodule Mesh, για να αυξήσουμε τον αριθμό των vertices του μοντέλου. Αυτό έχει ως αποτέλεσμα να διαιρεθεί η επιφάνεια του plane σε πολλά τετράγωνα. Οι πλευρές των μεγαλύτερων τετραγώνων (τα οποία περιέχουν μικρότερα μέσα τους) σχηματίζουν έναν σταυρό, το κέντρο του οποίου ταυτίζεται με το κέντρο του plane. Επίσης, βλέπουμε πως οι πλευρές των μικρότερων τετραγώνων σχηματίζουν σταυρούς, το κέντρο των οποίων ταυτίζεται με το vertex στο κέντρο των μεγαλύτερων τετραγώνων που τα περιλαμβάνουν (όπως φαίνεται και στην εικόνα 2). Εμείς με τις εντολές `....mountain.verts[21].co[2] += cliff1y * size...` ανυψώνουμε τα vertices αυτά για να σχηματίσουμε λόφους! Συγκεκριμένα, αναφερόμαστε στα εν λόγω vertices του μοντέλου, δίνοντας τους δείκτες τους στη λίστα `verts` (η οποία είναι μέλος του submodule Mesh). Επιπλέον, μέσω της `verts` διατρέχουμε τη λίστα `co[ ]` μέσω της οποίας μπορούμε να προσδιορίσουμε τη θέση του επιλεγμένου vertex σε οποιονδήποτε γεωμετρικό άξονα. Το `co[0]` αντιστοιχεί στον άξονα X, το `co[1]` στον Y και το `co[2]` στον Z (τον οποίο και χρησιμοποιούμε). Θέτουμε, λοιπόν, τη θέση (στον άξονα Z) των vertices, που σχηματίζουν κάθε λόφο, ίση με τη μετατόπιση που θέλουμε να τους δώσουμε επί το μέγεθος του μοντέλου. Η μετατόπιση για κάθε vertex αντιπροσωπεύεται από τις μεταβλητές `cliff1y`, `cliff2y`, `cliff3y` και `cliff4y`, ενώ το μέγεθος του μοντέλου αντιπροσωπεύει η μεταβλητή `size`. Στη συνέχεια, εξομαλύνουμε τις άκρες όλων των πολυγώνων του μοντέλου και τις κάνουμε πιο "απαλές", θέτοντας την ιδιότητα `smooth` των πλευρών του μοντέλου σε 1 (σαν να κάναμε κλικ στο `set smooth`). Μετά αυξάνουμε για άλλη μία φορά τα vertices του μοντέλου, κάνοντας `subdivide` άλλες δύο φορές. Αφού πλέον έχουμε αρκετά vertices, χρησιμοποιώντας έναν βρόχο για να τα προσπελάσουμε, δίνουμε στο καθένα μία τυχαία κατεύθυνση, θέτοντας τη θέση τους στους διάφορους γεωμετρικούς άξονες, ίση με το αποτέλεσμα της παράστασης `Rand(0.0,randomness) * size`. Η συνάρτηση `Rand(high=1,low=0)` που χρησιμοποιήσαμε γι' αυτή τη δουλειά, ανήκει στο submodule `mathutils` (που περιέχει μαθηματικές συναρτήσεις) και επιστρέφει μία τυχαία τιμή μεταξύ των τιμών `high` και `low`. Εμείς, λοιπόν,



Μετακινώντας αυτά τα vertices, φτιάχνουμε λόφους!

παίρνουμε μία τυχαία τιμή μεταξύ του 0 και της `randomness` και την πολλαπλασιάζουμε με το μέγεθος του μοντέλου. Με αυτό το τρόπο, δημιουργούμε τις αυλακώσεις του εδάφους...

## Εμμονή στη λεπτομέρεια...

Μέχρι τώρα, το σκριπτάκι μας μπορεί να φτιάξει το μοντέλο. Όμως, το αποτέλεσμα δεν δείχνει ρεαλιστικό, γιατί δεν έχουμε δώσει στο terrain έναν `subsurf modifier`, ο οποίος θα αυξήσει τον αριθμό των πολυγώνων του και θα στρογγυλέψει τις γωνίες. Για να το κάνετε αυτό, απλώς προσθέστε τον ακόλουθο κώδικα στη συνάρτηση:

```
mod=object.modifiers.append(Modifier.Types.SUBSURF)
mod[Modifier.Settings.LEVELS]=3
mod[Modifier.Settings.RENDLEVELS]=3
```

Έτσι, λοιπόν, δημιουργούμε έναν νέο `subsurf modifier` (μέσω του submodule `Modifier`) και τον προσθέτουμε στη λίστα με τους `modifiers` της οντότητας `object`. Μετά αποθηκεύουμε τη λίστα με τους `modifiers` του `object` στη λίστα `mod`. Με τις 2 επόμενες εντολές θέτουμε τον αριθμό των `levels` του `subsurf modifier` σε 3, τόσο στα `viewport` όσο και στο `rendering`.

## Δημιουργία GUI για το script

Επειδή το σκριπτάκι μας είναι αρκετά απλό, θα μπορούσε να δέχεται την είσοδο του χρήστη μέσω του τερματικού, όμως, είναι καλύτερη ιδέα να του φτιάξουμε ένα απλό γραφικό περιβάλλον (GUI) για να είναι πιο εύχρηστο. Προκειμένου να δημιουργήσουμε ένα GUI, θα χρησιμοποιήσουμε τα εργαλεία που μας προσφέρει το submodule `Draw` (ανήκει στο `Blender module`), για να φτιάξουμε τρεις συναρτήσεις. Η πρώτη θα ονομάζεται `event` και θα χειρίζεται τη γενικότερη είσοδο που μπορεί να δώσει ο χρήστης στο script μέσω των συσκευών εισόδου (ποντίκι, πληκτρολόγιο κ.λπ.) και τα σχετικά με αυτή συμβάντα. Η δεύτερη συνάρτηση θα έχει όνομα `button_event` και θα σχετίζεται με την αλληλεπίδραση του χρήστη με τα `widgets`, δηλαδή, θα χειρίζεται τα συμβάντα. Επιπλέον, η τρίτη συνάρτηση θα ονομάζεται `gui` και θα εμφανίζει τα `widgets` στην οθόνη. Τα `widgets` του GUI που θα κατασκευάσουμε θα είναι 5 `Number sliders` (`sliders`, δηλαδή, στα οποία εμφανίζεται μία αριθμητική τιμή) και 2 `κουμπιά`. Τα 5 `sliders` θα αφορούν στις διαστάσεις και στη μορφή του μοντέλου, ενώ με τα 2 `κουμπιά` ο χρήστης θα μπορεί να σταματά την εκτέλεση του script και να δημιουργεί το terrain. Το τελευταίο βήμα που πρέπει να κάνουμε για την κατασκευή του GUI, είναι η κλήση της συνάρτησης `Register` (μέλος του `Draw`), για να καθορίσουμε τις εργασίες για τις οποίες θα είναι υπεύθυνες οι τρεις παραπάνω συναρτήσεις (που θα είναι παράμετροι στη `Register`). Ας δούμε, όμως, πώς θα υλοποιήσουμε όλα τα παραπάνω, δημιουργώντας τις απαραίτητες μεταβλητές (στο `global namespace`):

```
#Τα sliders που καθορίζουν τη μορφή του εδάφους
num1=Draw.Create(0.1)
num2=Draw.Create(0.1)
num3=Draw.Create(0.1)
num4=Draw.Create(0.1)
#Το slider που καθορίζει την τραχύτητα του εδάφους
numrandom=Draw.Create(0.08)
#Τα κουμπιά τερματισμού και δημιουργίας του terrain
btncreate=0
btnexit=0
evtnum1=0
evtnum2=1
evtnum3=2
evtnum4=3
evtrandom=4
evtcreate=5
evtexit=6
```

Αρχικά, λοιπόν, δημιουργούμε τα `widgets` ως στοιχεία του GUI,

χρησιμοποιώντας τη συνάρτηση Create του submodule Draw . Η παράμετρος που δέχεται η εν λόγω συνάρτηση, είναι η αρχική τιμή του widget, η οποία μπορεί να είναι τύπου string, integer ή float. Επειτα, στις μεταβλητές που ακολουθούν, των οποίων η ονοματολογία είναι του τύπου εντόνομα \_widget, θέτουμε τον αριθμό συμβάντος για κάθε widget. Ο αριθμός αυτός χρησιμοποιείται στην button\_event για να ανιληφθεί το script με ποιο widget αλληλεπιδρά ο χρήστης. Ας υλοποιήσουμε τώρα, όμως, τη συνάρτηση event:

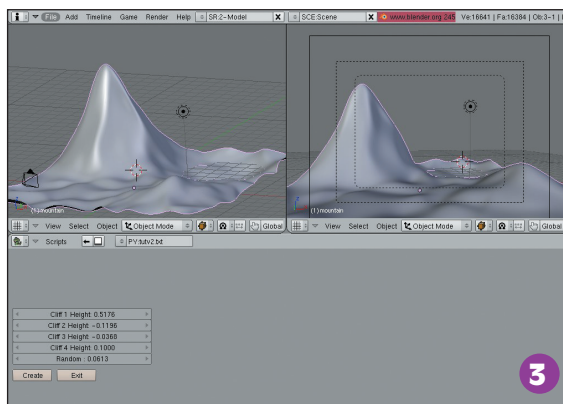
```
def event(evt, val):
    if evt == Draw.ESCKEY:
        Draw.Exit()
```

Σε αυτή τη συνάρτηση, λοιπόν, το μόνο που κάνουμε, είναι να ελέγχουμε πότε ο χρήστης πατά το πλήκτρο Escape (απλώς ελέγχουμε πότε η μεταβλητή evt είναι ίση με τη σταθερά ESCKEY, και σε αυτή τη περίπτωση το σκριντάκι θα τερματίζεται μέσω της κλήσης Draw.Exit()). Θα προσέξατε πως ως παράμετρο στη συνάρτηση έχουμε και τη μεταβλητή val, η οποία, όταν ισούται με 1, μας ενημερώνει πως ένα πλήκτρο είναι πατημένο και, όταν είναι ίση με 0, πως ένα πλήκτρο απελευθερώνεται (συμπεριλαμβανομένων των κλικ του ποντικιού). Παρόλο που δεν τη χρησιμοποιούμε στο script, πρέπει να υπάρχει στις παραμέτρους της συγκεκριμένης συνάρτησης, γιατί σε αντίθετη περίπτωση θα συναντήσουμε πρόβλημα στο να δώσουμε την event ως παράμετρο στη συνάρτηση Draw.Register(). Ας δούμε, όμως, πως μπορούμε να χειριστούμε τα events των widget μέσω της συνάρτησης button\_event:

```
def button_event(evt):
    global evtnum1, evtnum2, evtnum3,
    evtnum4, evtrandom, evtcreate, evtexit
    if evt == evtcreate:
        createterrain(5, num1.val, num2.val, num3.val, num4.val,
            numrandom.val)
    if evt == evtexit:
        Draw.Exit()
```

Εδώ, λοιπόν, ελέγχουμε πότε ο χρήστης θα κλικάρει τα κουμπιά btncreate και btnexit. Με το πρώτο κουμπί θα δημιουργείται το terrain. Γι' αυτό, λοιπόν, όταν συμβαίνει το evtcreate, εκτελείται η συνάρτηση createterrain, στην οποία δίνουμε ως ορίσματα το μέγεθος του μοντέλου (5), τις αριθμητικές τιμές των number sliders (num1.val κ.λπ.) που καθορίζουν τη μορφή του terrain και, τέλος, την αριθμητική τιμή του slider, που καθορίζει το πόσο ανώμαλη θα είναι η επιφάνεια του εδάφους (numrandom.val). Στη περίπτωση, όμως, που ο χρήστης κάνει κλικ στο κουμπί btnexit, τότε καλείται η συνάρτηση Exit() (από το submodule Draw) και το σκριντάκι μας τερματίζεται. Ας περάσουμε, όμως, σιγά-σιγά και στο σχεδιασμό του GUI:

```
def gui():
    global num1, num2, num3, num4,
    numrandom, btncreate, btnexit
    global evtnum1, evtnum2, evtnum3,
    evtnum4, evtrandom, evtcreate, evtexit
    #δημιουργία των number sliders
    num1=Draw.Number("Hill 1
    Height:", evtnum1, 10, 160, 250, 20, num1.val, -0.9, 0.9,
    "Enter a value")
    num2=Draw.Number("Hill 2
    Height:", evtnum2, 10, 140, 250, 20, num2.val, -0.9, 0.9,
    "Enter a value")
    num3=Draw.Number("Hill 3
    Height:", evtnum3, 10, 120, 250, 20, num3.val, -0.9, 0.9,
    "Enter a value")
    num4=Draw.Number("Hill 4
    Height:", evtnum4, 10, 100, 250, 20, num4.val, -0.9, 0.9,
    "Enter a value")
    numrandom=Draw.Number("Random:
```



Το script μας σε δράση: μόλις δημιουργήσαμε το πρώτο terrain, χρησιμοποιώντας το!

```
", evtrandom, 10, 80, 250, 20, numrandom.val, 0.01, 0.12,
"Enter a value")
```

```
#δημιουργία των κουμπιών
```

```
btncreate=Draw.Button("Create", evtcreate, 10, 50, 70, 20,
"Click to Create Terrain")
```

```
btnexit=Draw.Button("Exit", evtexit, 90, 50, 70, 20,
"Click to Exit")
```

Στη συνάρτηση gui, αρχικά ενημερώνουμε την Python πως έχουμε δηλώσει τα widgets και τα events τους ως global μεταβλητές. Μετά δημιουργούμε τα Number Sliders και εκχωρούμε το καθένα στη μεταβλητή του. Αυτό το επιτυγχάνουμε με τη συνάρτηση Number() η οποία είναι μέλος του submodule Draw. Η εν λόγω συνάρτηση δέχεται ως ορίσματα το όνομα του slider, τον αριθμό συμβάντος του slider, τις θέσεις x και y στο παράθυρο, το πλάτος, το ύψος, την τρέχουσα τιμή του, την ελάχιστη και μέγιστη τιμή που μπορεί να πάρει και, τέλος, το tooltip του widget. Αφού, λοιπόν, τελειώσουμε με τα Number sliders, συνεχίζουμε με τα κουμπιά. Εκεί τα πράγματα είναι ελαφρώς διαφορετικά. Αυτή τη φορά, όμως, καλούμε τη συνάρτηση Button() (μέλος του Draw) και της δίνουμε ως ορίσματα μόνο το όνομα, τον αριθμό συμβάντος, τις θέσεις x και y, το πλάτος και το ύψος και, τέλος, απλώς το tooltip. Για την ολοκλήρωση του script μας, μένει πλέον μόνο ένα βήμα: η κλήση της συνάρτησης Draw.Register() για να καθορίσουμε για ποια λειτουργία θα είναι υπεύθυνη η καθεμία από τις τρεις παραπάνω συναρτήσεις:

```
Draw.Register(gui, event, button_event)
```

Μπορείτε να δείτε σε αυτή τη γραμμή κώδικα πως κατά την κλήση της Draw.Register(), της δώσαμε ως παραμέτρους τις συναρτήσεις gui, event και button\_event. Η πρώτη παράμετρος (στη περίπτωση μας gui) αντιστοιχεί στη συνάρτηση που ζωγραφίζει τα widgets, η δεύτερη (event) στη συνάρτηση που θα χειρίζεται τα συμβάντα εισόδου από συσκευές όπως το ποντίκι και το πληκτρολόγιο και, τέλος (button\_event), ακολουθεί η συνάρτηση χειρισμού των συμβάντων των widgets.

Τώρα, λοιπόν, που ολοκληρώσαμε τη συγγραφή του script, ήρθε η ώρα να το δοκιμάσουμε. Από το μενού File του κειμενογράφου πατήστε τη Run Python Script, για να το εκτελέσετε. Αμέσως μόλις ξεκινήσει, θα πρέπει να δείτε το γραφικό περιβάλλον του. Δώστε τις τιμές που θέλετε στα sliders που αφορούν στη μορφή του εδάφους (Hill1, Hill2 κ.λπ.) και επίσης καθορίστε το πόσο τραχιά θα δείχνει η επιφάνεια μέσω του slider, με το όνομα random. Μετά πατήστε το κουμπί create για να δημιουργηθεί το terrain, με βάση τις ρυθμίσεις που κάνατε. Αμέσως, το μοντέλο του terrain θα ξεπροβάλει στο παράθυρο 3dview (όπως στην εικόνα 3). Αν θέλετε να τερματίσετε το σκριντάκι, μπορείτε να πατήσετε το πλήκτρο Escape ή να κλικάρετε το κουμπί Exit. Το μόνο που του λείπει πλέον, είναι ένα υλικό. Αλλά αυτό θα πρέπει να το προσθέσετε μόνοι σας χειροκίνητα με βάση τις γνώσεις που έχετε αποκτήσει από τα προηγούμενα άρθρα της σειράς. Μην σας τα λέω όλα πάλι...

ΤΗΝ  
ΕΠΟΜΕΝΗ  
ΦΟΡΑ

Παιχνίδια με την  
game engine!