1. *RigeN*

U ovom tekstu objašnjena je ideja skripte pod nazivom *RigeN*. Ona se sastoji iz dva dela: *RigeN_Armature* i *RigeN_Constraints.* Implementirana je u *OpenSource* softveru *Blender* u ovkiru *Text Editor* prozora i pisana je u programskom jeziku *Python*.

Cilj je da se ubrza vreme produkcije riginga jednostavnih karaktera za animaciju kod kratkometražnih animiranih filmova i igara.

1. *RigeN Armature*

Prva skripta *Rigen_Armature* polazi od postavke kostiju nogu i torza. U njih se ubrajaju sledeće kosti:

Torso, Thigh.L, Shin.L, Foot.L, Toe.L, Heel,L, Thigh.R, Shin.R, Foot.R, Toe.R, Heel,R.

Blender svaki objekat, u ovom slucaju digitalni skelet, predstavlja kroz klasu *Object* koja ima svoju potklasu *Armature,* stoga je bilo potrebno napisati funkciju kojom će se kreirati objekat *RigeN* koji će imati svoju armaturu *RigeN.Armature*. Objekat je postavljen u (0, 0, 0) koordinatnom sistemu.

```
def createArmature(origin):
    # Create armature and object
    amt = bpy.data.armatures.new('RigeN.Armature')
    rig = bpy.data.objects.new('RigeN', amt)
    rig.location = origin
    rig.show_x_ray = True
    amt.show_names = True
    # Link object to scene
    scn = bpy.context.scene
    scn.objects.link(rig)
    scn.objects.active = rig
    scn.update()
```

Slika 1.1 Kreiranje armature

Svakom objektu sa potklasom *Armature* je moguće pristupiti iz 3 modaliteta: *Object Mode, Pose Mode* i *Edit Mode*. Kako bi se kreirale kosti za postavljenu armaturu, potrebno je pristupiti *Edit Mode*-u.

```
    bpy.ops.object.editmode_toggle()
    bpy.context.object.data.use_mirror_x = True
```

Slika 1.2 *Edit Mode*

Omogućen je pristup korigovanja armature sa simetrijom oko X ose.

Objekat za kost je definisan pozicijom glave i repa, hijerarhijskim relacijama sa drugim kostima iz skeleta (*child - parent*) i izborom da li će kost uticati na deformaciju modela karaktera ili ne (*use_deform*) (slika 1.3).

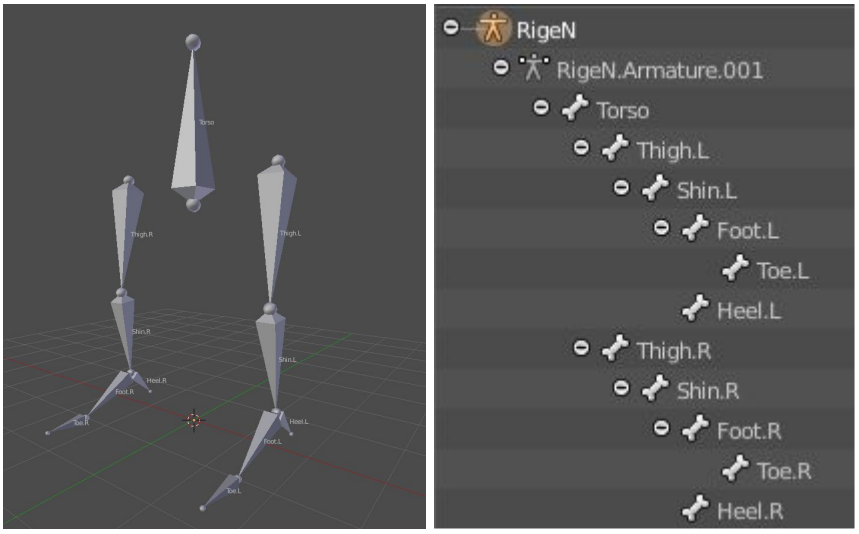
```
torso = amt.edit_bones.new('Torso')
torso.head = (0, 0, 3.5)
torso.tail = (0, 0, 6)
torso.use_deform = False

### LEFT LEG ###
thighL = amt.edit_bones.new('Thigh.L')
thighL.head = (1.5, 0.0, 4.0)
thighL.tail = (1.5, -0.2, 2.0)
thighL.parent = torso
thighL.use_connect = False
```

Slika 1.3 Kreiranje kostiju

Za postavljene kosti je moguće ručno menjati poziciju, orijentaciju i veličinu kako bi se skelet prilagodio anatomiji svakog modela.

Hijerarhijska struktura postavljenih kostiju trenutno izgleda kao na slici 1.4.



Slika 1.4 Hijerarhijska struktura postavljenih kostiju

RigeN Constraints

Drugi deo skripte se odnosi na postavljanje veza između postavljenih kostiju i kreiranje novih kostiju koje ce služiti kao kontrole za animaciju. Takve veze u 3D softverima se nazivaju *Constraints*.

Prvo je bilo neophodno u novu *Object* klasu upisati prethodno kreirani *RigeN*, a takođe u novu *Armature* klasu upisati *Rigen.Armature*.

```
rig = bpy.data.objects['RigeN']
rig.data.name = "RigeN.Armature"
amt = bpy.data.armatures['RigeN.Armature']

scn = bpy.context.scene
scn.objects.active = rig
scn.update()
```

Slika 1.5 Upisivanje *RigeN* ojbekta

U okviru *Edit Mode* selekcije, upisane su kreirane kosti u nove promenljive - istog imena kako bi konvencija bila očuvana i kako bi dalji kod u skripti bio jasno čitljiv. Za kreiranje novih kostiju poslužili su parametri upisanih kostiju - pozicije glave i repa.

```
### Ucitane kosti iz EDIT MODA

torso = amt.edit_bones['Torso']

####### CREATE_BONE_HELPERS FOT SPINE
spine = amt.edit_bones.new('Spine')
spine.head = torso.head
spine.tail = torso.tail
spine.parent = torso
spine.roll = torso.roll

spineCtrl = amt.edit_bones.new('Spine.Ctrl')
spineCtrl.head = spine.tail
spineCtrl.tail = spine.tail + (spine.tail - spine.head)/4.0
spineCtrl.parent = torso
```

Slika 1.6 Kreiranje novih kostiju za potrebu kontrola

```
####### CREATE LEFT FOOT CONTROL #######
footCtrlLeft = amt.edit_bones.new('Foot.Ctrl.L')
footCtrlLeft.head = ikStartL
footCtrlLeft.tail = (ikEndL.x, ikEndL.y, ikStartL.z)
footCtrlLeft.parent = None
footCtrlLeft.use_connect = False

####### CREATE RIGHT FOOT CONTROL #######
footCtrlRight = amt.edit_bones.new('Foot.Ctrl.R')
footCtrlRight.head = ikStartR
footCtrlRight.tail = (ikEndR.x, ikEndR.y, ikStartR.z)
footCtrlRight.parent = None
footCtrlRight.use_connect = False

####### CREATE LEFT HEEL CONTROL #######
heelRollLeft = amt.edit_bones.new('Heel.Roll.L')
heelRollLeft.head = heelL.tail
heelRollLeft.tail = rollHelperL
heelRollLeft.parent = footCtrlLeft
heelRollLeft.use_connect = False

####### CREATE RIGHT HEEL CONTROL #######
heelRollRight = amt.edit_bones.new('Heel.Roll.R')
heelRollRight.head = heelR.tail
heelRollRight.tail = rollHelperR
heelRollRight.parent = footCtrlRight
heelRollRight.use_connect = False

####### CREATE LEFT TOE CONTROL #######
toeRollLeft = amt.edit_bones.new('Toe.Roll.L')
toeRollLeft.head = footL.tail
toeRollLeft.tail = rollHelperL
toeRollLeft.parent = heelRollLeft
```

Slika 1.7 Kreiranje ostalih kontrola

Nakon što su sve potrebne kontrole kreirane, uspostavljena je relacija sa starim kostima, odnosno postavljene su *constraint* veze. Bilo je potrebno pristupiti *Pose Mode* selekciji kostiju. U nove promenljive, sa sufiksom *-Pose* upisani su prehodno kreirane na koje će se dalje postaviti *constraint* veze a sufiks *-Ctrl* imaju kosti koje će biti kontre za animaciju. Videti na slici 1.8.

```
bpy.ops.object.mode_set(mode='POSE')

footCtrlL = rig.pose.bones['Foot.Ctrl.L']
footCtrlL.rotation_mode = 'XYZ'

torsoPose = rig.pose.bones['Torso']

spinePose = rig.pose.bones['Spine']

spineCtrlPose = rig.pose.bones['Spine.Ctrl']
spineCtrlPose.lock_location[0] = True
spineCtrlPose.lock_location[2] = True
spineCtrlPose.lock_rotation[0] = True # x
spineCtrlPose.lock_rotation[2] = True # z
```

Slika 1.8 *Pose Mode*

Na slici 1.8 je takođe prikazano da su za neke kosti zaključane ose rotacije ili lokacije (translacije) - što znači da one služe jedino za translaciju i rotaciju oko jedne ose.

U daljem tekstu biće prikazani samo neki od postavljenih *constraint* veza:

```
toeRollLeft = rig.pose.bones['Toe.Roll.L']
toeCopRotL = toeRollLeft.constraints.new('COPY_ROTATION')
toeCopRotL.name = 'Toe.Copy.Rotation.L'
toeCopRotL.target = rig
toeCopRotL.subtarget = 'Foot.Roll.L'
toeCopRotL.owner_space = 'LOCAL'
toeCopRotL.target_space = 'LOCAL'
toeCopRotL.use_y = False
toeCopRotL.use_z = False
```

Slika 1.9 *Copy Rotation Constraint*

Iz slike 1.9 se vidi da je *constraint Copy Rotation* postavljen na kost *Toe.Roll.L* i da je nazvan *'Toe.Copy.Rotation.L'*. Njegov target je *Object RigeN* sa konkretnim targetom na *'Foot.Roll.L'*. Kopiraće rotaciju targetne kosti ali u svom lokalnom koordinatnom sistemu obzirom da su ove dve kosti međusobno različito orijentisane. *False* tvrdnja za izbor osa rotacija je označila da se neće koristiti rotacija oko Y i Z ose, već samo oko X ose.

```
shinR = rig.pose.bones['Shin.R']
IKRightFoot = shinR.constraints.new('IK')
IKRightFoot.name = 'IK.R'
IKRightFoot.target = rig
IKRightFoot.subtarget = 'Foot.IK.R'
IKRightFoot.chain_count = 2
```

Slika 1.10 *IK Constraint*

Na slici 1.10 prikazano je dodavanje *constraint IK*. *IK* je skraćenica od *Inverse Kinematics* i gradi takvu hijerarhijsku strukturu da poslednja član u lancu utiče na poziciju i orijentaciju svih prethodnih članova. Drugim rečima, kost za stopalo će svojim pomeranjem uticati na poziciju i orijentaciju kostiju potkolenice i natkolenice. Parametar *chain_count* = 2 govori da će u *IK* lancu biti svega dve kosti koje će trpeti ovakve transformacije.
Na samom kraju, nakon svih dodatih *constraint* veza, napravljene su liste kostiju za koje je bilo potrebno zaključati mogućnosti skaliranja, rotiranja i transliranja.

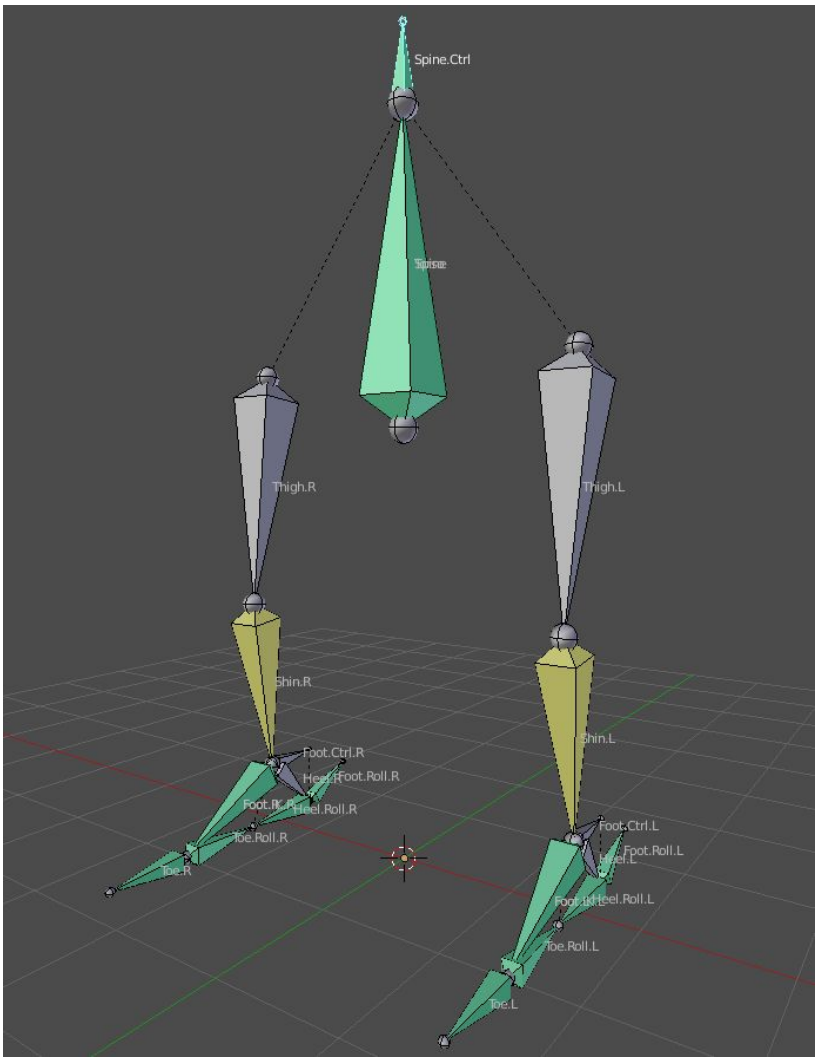
```
### SCALE LOCK
bone_list_lock_scale = ['Foot.L', 'Heel.L', 'Toe.L',
'Shin.L', 'Thigh.L', 'Foot.Roll.L','Foot.Ctrl.L',
'Foot.IK.L', 'Toe.Roll.L', 'Heel.Roll.L', 'Foot.Roll.L',
'Foot.R', 'Heel.R', 'Toe.R', 'Shin.R', 'Thigh.R',
'Foot.Roll.R','Foot.Ctrl.R', 'Foot.IK.R', 'Toe.Roll.R',
'Heel.Roll.R', 'Foot.Roll.R', 'Torso', 'Spine.Ctrl']
for bn in bone_list_lock_scale :
    bone = bpy.context.active_object.pose.bones[bn]
    bone.lock_scale[0] = True # x
    bone.lock_scale[1] = True # y
    bone.lock_scale[2] = True # z

### ROTATION LOCK
bone_list_lock_rotation = ['Thigh.R', 'Thigh.L']
for bn in bone_list_lock_rotation :
    bone = bpy.context.active_object.pose.bones[bn]
    bone.lock_rotation[0] = True # x
    bone.lock_rotation[1] = True # y
    bone.lock_rotation[2] = True # z

### LOCATION LOCK
bone_list_lock_location = ['Foot.Roll.L', 'Foot.Roll.R']
for bn in bone_list_lock_location :
    bone = bpy.context.active_object.pose.bones[bn]
    bone.lock_location[0] = True # x
    bone.lock_location[1] = True # y
    bone.lock_location[2] = True # z
```

Slika 1.11 Zaključavanje transformacija

FInalni rezultat se može videti na slici 1.12, gde su kosti na kojima su postavljene *constraint* veze obojene zelenom ili žutom bojom.



Slika 1.12 *RigeN*