Pour ajouter une fonctionnalité de simplification d'un objet 3D dans ton projet en utilisant Django REST Framework (DRF), voici les étapes à suivre :

1. Installer Django REST Framework

Si ce n'est pas déjà fait, tu dois installer Django REST Framework. Ajoute-le à ton fichier `requirements.txt` ou installe-le directement via pip :

```
pip install djangorestframework
```

Ensuite, ajoute `rest_framework` à la liste des applications installées dans ton fichier `settings.py` :

```
INSTALLED_APPS = [
    # autres applications
    'rest_framework',
]
```

2. Créer une API pour la Simplification 3D

a. Créer un Serializer

Les serializers permettent de convertir les instances de ton modèle en formats JSON ou XML, et vice versa.

Crée un fichier `serializers.py` dans ton dossier `processing` (si ce n'est pas déjà fait) et définis un serializer pour ton modèle 3D :

```
# backend/processing/serializers.py
from rest_framework import serializers
from .models import Model3D  # Assure-toi d'importer le modèle adéquat

class Model3DSerializer(serializers.ModelSerializer):
    class Meta:
        model = Model3D
        fields = '__all__'
```

b. Créer une Vue API

Crée une vue qui utilisera le serializer pour gérer les requêtes HTTP. Tu peux utiliser les vues génériques de DRF pour simplifier cela.

Ajoute une nouvelle vue dans `views.py` :

```python
# backend/processing/views.py
from rest_framework import viewsets
from .models import Model3D
from .serializers import Model3DSerializer

class Model3DViewSet(viewsets.ModelViewSet):
    queryset = Model3D.objects.all()
    serializer_class = Model3DSerializer

    def perform_create(self, serializer):
        # Ajoute ici ta logique de simplification 3D
        model3d = serializer.save()
        self.simplify_model(model3d)

    def simplify_model(self, model3d):
        # Implémente ta logique de simplification ici
        pass
```

c. Configurer les Routes API

Définis les routes pour ton API en ajoutant une configuration dans `urls.py` :

```python
# backend/processing/urls.py
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import Model3DViewSet

router = DefaultRouter()
router.register(r'models3d', Model3DViewSet)

urlpatterns = [
    path('', include(router.urls)),
]
```

3. Configurer la Logique de Simplification

Dans la méthode `simplify_model` de `Model3DViewSet`, implémente la logique nécessaire pour simplifier l'objet 3D. Cela peut inclure des opérations comme réduire le nombre de polygones ou appliquer des algorithmes de simplification géométrique.

4. Tester l'API

Une fois que tout est configuré, tu peux tester ton API en utilisant un client REST comme Postman ou cURL. Assure-toi que les requêtes GET, POST, PUT, et DELETE fonctionnent correctement et que la logique de simplification est appliquée comme prévu.

5. **Documenter l'API**

Enfin, tu peux ajouter une documentation pour ton API en utilisant des outils comme Swagger ou ReDoc intégrés dans DRF. Assure-toi que les utilisateurs de ton API comprennent comment utiliser les points de terminaison disponibles.

Exemple de Test avec cURL

Pour tester la création d'un modèle 3D via l'API :

```
curl -X POST http://localhost:8000/models3d/ -H "Content-Type: application/json" -d '{"field1": "value1", "field2": "value2"}'
```

Cela enverra une requête POST à ton API pour créer un nouveau modèle 3D avec les données fournies, en appliquant la logique de simplification définie dans `simplify_model`.

Assure-toi de remplacer `field1`, `field2`, etc., par les champs réels de ton modèle 3D.