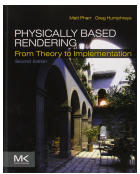
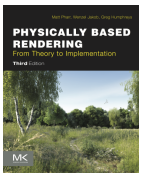

LUND UNIVERSITY
LTH
FACULTY OF ENGINEERING
PBRT: Create your own Importers and Exporters
Gustaf Waldemarson
Arm & Lund University
BCON24
BLENDER CONFERENCE
OCTOBER 23 - 24 - 25
AMSTERDAM

Who am I?

- Industrial PhD Student at the Lund University Graphics Group
- Software Engineer at Arm

arm





https://gustafwaldemarson.com

ECGN24

LUND UNIVERSITY

Agenda

1. What is PBRT?
   - Why use it?
2. The Importer
   - Earlier Work
3. Proxy Objects and Renderer Settings
4. The Exporter

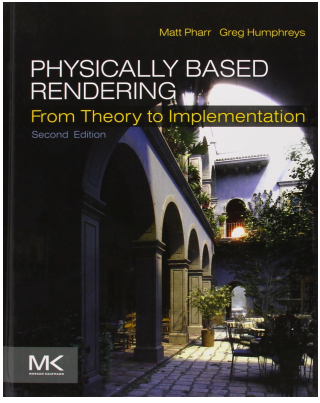
Blender Renderers

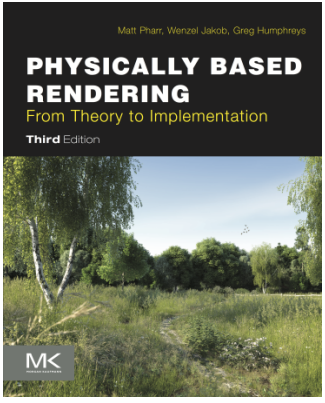
Blender

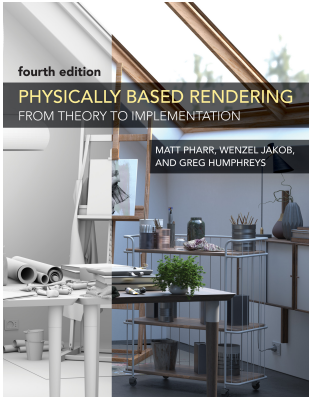
Cycles	Unidirectional volumetric path tracer
Eevee	Real-time *rasterization* based rendering engine
Workbench	Real-time *preview* rendering engine

PBRT

Physically Based Rendering – From Theory to Implementation







https://pbr-book.org/3ed-2018/contents
https://pbr-book.org/4ed/contents

PBRT – The Renderer(s)

PBRT-v2

- Ambient Occlusion
- Instant Global Illumination
- Irradiance Cache
- Path Tracer
- Light Probes
- Whitted RT
- …

PBRT-v3

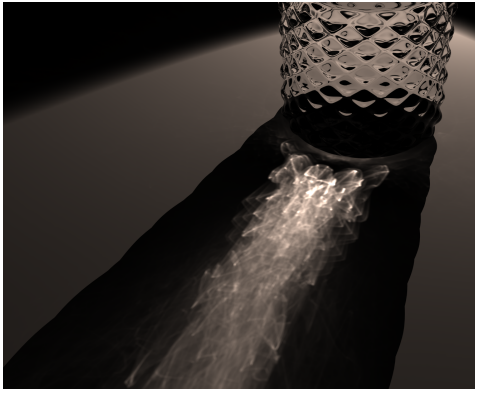
- Bidirectional Path Tracing
- Direct Lighting
- Whitted RT
- Path Tracing
- Volumetric Path Tracing
- Metropolis Light Transport
- Stochastic Progressive Photon Mapping
- …

PBRT-v4

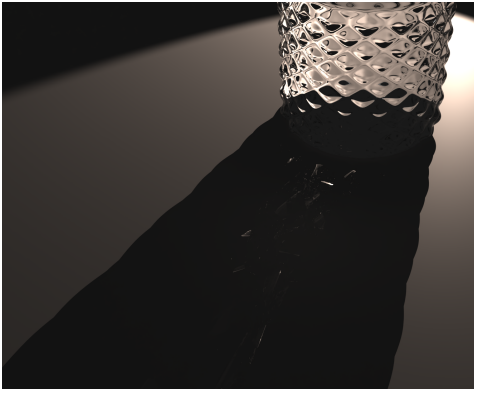
- Bidirectional Path Tracing
- Direct Lighting
- Whitted RT
- (Volumetric) Path Tracing
- Metropolis Light Transport
- Light Path Tracing
- Simple (Volumetric) Path Tracing
- Stochastic Progressive Photon Mapping
- …

PBRT – Why?

Photon Mapping Example



PBRTv3 - SPPM



Cycles

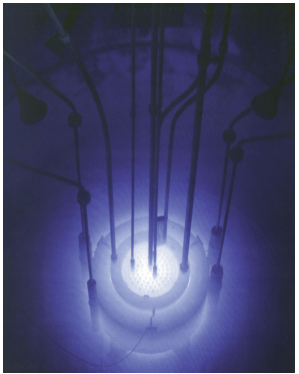
Model courtesy of Simon Wendsche

HBO Miniseries: Chernobyl

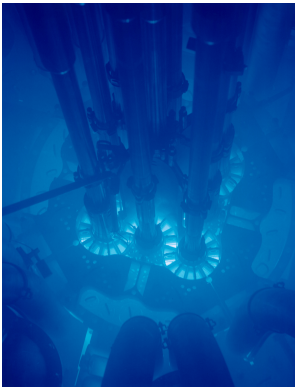
Cherenkov Radiation

– Images from “Chernobyl” Episode 1-2 © HBO

Nuclear Reactors



Reed Research Reactor

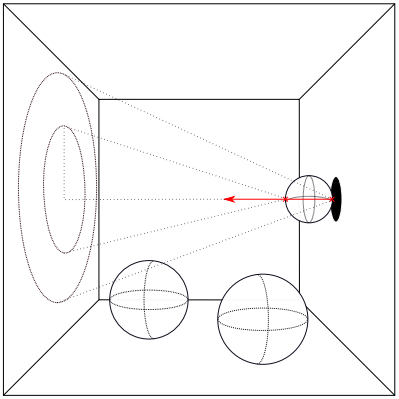


Advanced Test Reactor

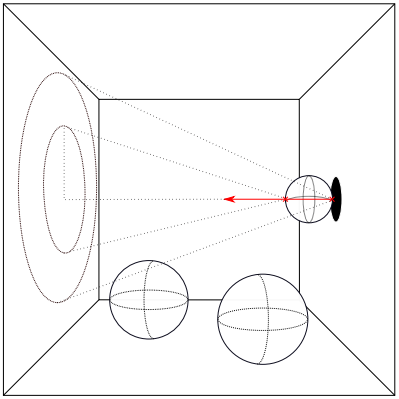
Superluminal Photon Mapping

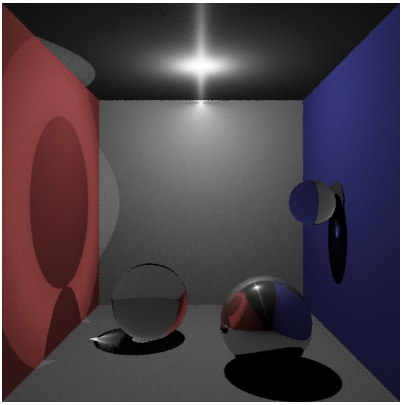
Cherenkov Radiation

Superluminal Photon Mapping

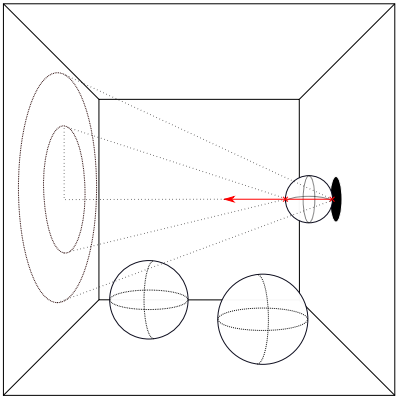


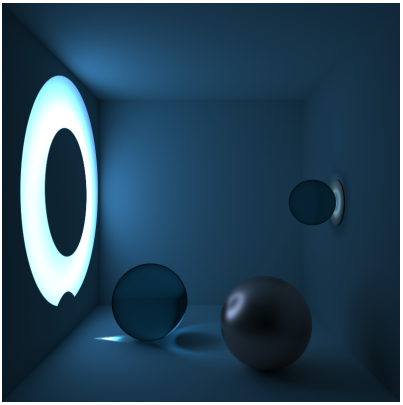
Superluminal Photon Mapping



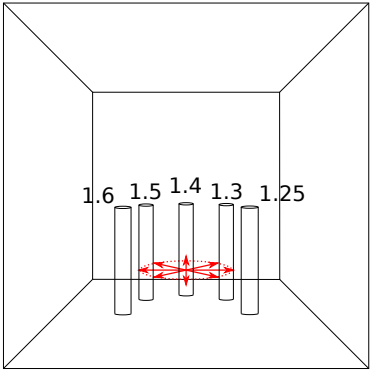


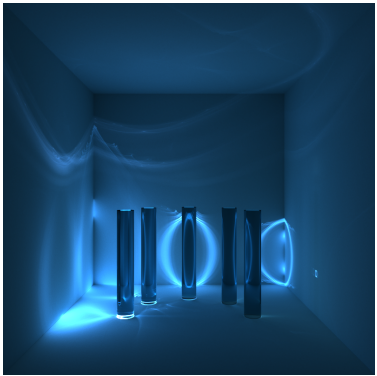
Superluminal Photon Mapping





Superluminal Photon Mapping





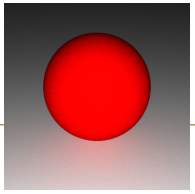
Gustaf Waldemarson and Michael Doggett. "Photon Mapping Superluminal Particles". In: *Eurographics 2020 - Short Papers*. Ed. by Alexander Wilkie and Francesco Banterle. The Eurographics Association, 2020. ISBN: 978-3-03868-101-4. DOI: 10.2312/egs.20201004

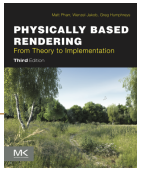

blender™
fourth edition
PHYSICALLY BASED RENDERING
FROM THEORY TO IMPLEMENTATION
MATT PHARR, WENZEL JAKOB, AND GREG HUMPHREYS
Matt Pharr, Wenzel Jakob, Greg Humphreys
PHYSICALLY BASED RENDERING
From Theory to Implementation
Third Edition
Matt Pharr Greg Humphreys
PHYSICALLY BASED RENDERING
From Theory to Implementation
Second Edition
LUGG
Lund University Graphics Group

The Importer

The Format

The Importer — sphere-ex.pbrt

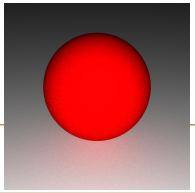


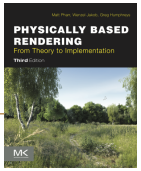



```
LookAt 0 5 8  0 .8 0   0 1 0
Film "image"
     "integer xresolution" [400] "integer yresolution" [400]
     "string filename" "sphere-ex.exr"
Camera "perspective" "float fov" [22]
Integrator "path"
WorldBegin
    AttributeBegin
        CoordSysTransform "camera"
        LightSource "point" "color I" [300 300 300]
    AttributeEnd
    Include "mesh.pbrt"
    Include "sphere.pbrt.gz"
WorldEnd
```

The Format

The Importer — mesh.pbrt / sphere.pbrt.gz





mesh.pbrt

```
AttributeBegin
    Material "matte" "color Kd" [.8 .8 .8 ]
    Shape "trianglemesh" "integer indices" [0 2 1 2 0 3]
          "point P" [-10 0 -10   10 0 -10   10 0 10   -10 0 10 ]
AttributeEnd
```

sphere.pbrt.gz

```
AttributeBegin
    Translate 0 1 0
    Rotate 60 1 1 1
    Material "uber" "color Kd" [.8 .0 .0] "color Ks" [.05 .05 .05]
    Shape "sphere"
AttributeEnd
```

Earlier Work

The Importer

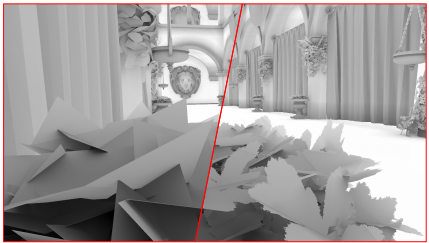
- https://github.com/mxpv/pbrt4 (Rust)
- https://github.com/vilya/minipbrt (C++)
- https://github.com/ingowald/pbrt-parser (C++)

An Earlier Attempt: PBRT → glTF → Blender

The Importer

Issues

- Most PBRT objects cannot be represented
- We lose the material and texture graphs



The Importer

What do we need to do?

- An Import *Operator*[1]
- An Import Panel[2] (optional)
- An Import Menu Entry[3] (optional)

[1] https://docs.blender.org/manual/en/latest/interface/operators.html
[2] https://docs.blender.org/api/current/bpy.types.Panel.html
[3] https://docs.blender.org/api/current/bpy.types.Menu.html

The Operator

The Importer


```
class ImportPBRT(bpy.types.Operator, bpy_extras.io_utils.ImportHelper):
    """Blender Importer class for PBRT scenes."""
    # ... Config variables...

    def draw(self, context):
        """Specify how to draw the Blender panel UI."""

    def execute(self, context):
        """Execute the PBRT import process."""
```

The Operator

The Importer – Configuration Variables


```
class ImportPBRT(bpy.types.Operator, bpy_extras.io_utils.ImportHelper):
    """Blender Importer class for PBRT scenes."""
    bl_idname = "import_scene.pbrt"
    bl_label = "Import PBRT"
    bl_options = {"REGISTER", "UNDO"}

    filter_glob: StringProperty(default="*.pbrt;*.pbrt.gz")
    files: CollectionProperty(name="File Path",
                              type=bpy.types.OperatorFileListElement)
    mode: EnumProperty()
    parse_only: BoolProperty()
```

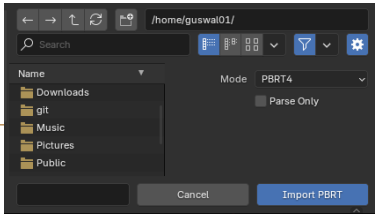
The Operator

The Importer – Configuration Variables


```
class ImportPBRT(bpy.types.Operator, bpy_extras.io_utils.ImportHelper):
    """Blender Importer class for PBRT scenes."""
    bl_idname = "import_scene.pbrt"
    bl_label = "Import PBRT"
    bl_options = {"REGISTER", "UNDO"}

    filter_glob: StringProperty(default="*.pbrt;*.pbrt.gz")
    files: CollectionProperty(name="File Path",
                              type=bpy.types.OperatorFileListElement)
    mode: EnumProperty()
    parse_only: BoolProperty()
```

The Panel

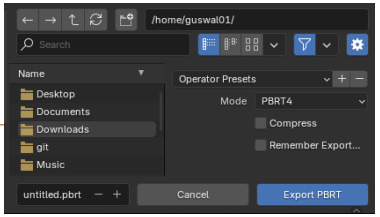
The Importer




```
def draw(self, context):
    """Specify how to draw the Blender panel UI."""
    sfile = context.space_data
    operator = sfile.active_operator
    self.layout.use_property_split = True
    self.layout.use_property_decorate = False
    self.layout.prop(operator, "mode")
    self.layout.prop(operator, "parse_only")
```

The Panel in a Separate Class

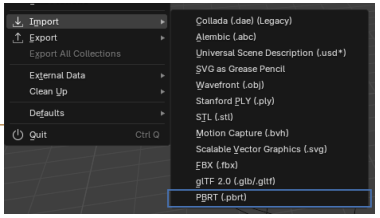
The Exporter Panel




```
class PBRT_PT_export(bpy.types.Panel):
    bl_space_type = "FILE_BROWSER"
    bl_region_type = "TOOL_PROPS"
    bl_label = ""
    bl_parent_id = "FILE_PT_operator"
    bl_options = {"HIDE_HEADER"}

    @classmethod
    def poll(cls, context):
        sfile = context.space_data
        operator = sfile.active_operator
        return operator.bl_idname == "EXPORT_SCENE_OT_pbrt"

    def draw(self, context):
        # ...
```

The Menu Entry

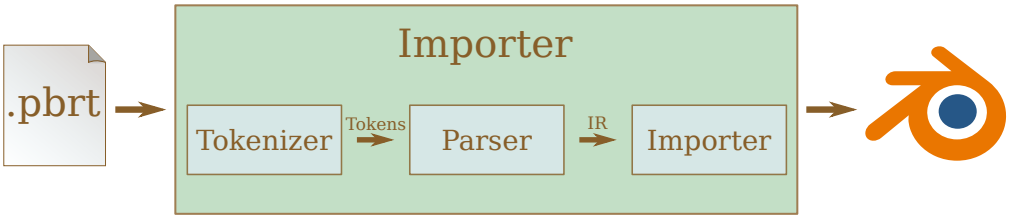
The Importer




```
def menu_func_import(self, context):
    """Function to run when executing the import menu item."""
    self.layout.operator(ImportPBRT.bl_idname, text='PBRT (.pbrt)')

def register():
    """Register the addon with Blender."""
    bpy.types.TOPBAR_MT_file_import.append(menu_func_import)

    # ...
```

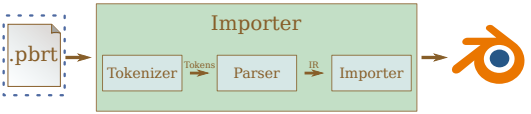
Parsing

The Importer



The Operator

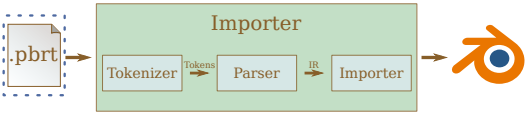
The Importer – `execute()`




```python
def execute(self, context):
    """Execute the PBRT import process."""
    settings = self.as_keywords()
    # Multiple files?
    if self.files:
        dirname = os.path.dirname(self.filepath)
        for file in self.files:
            self.unit_import(os.path.join(dirname, file.name), settings)
        return {'FINISHED'}
    else:
        return self.unit_import(self.filepath, settings)
```

The Operator

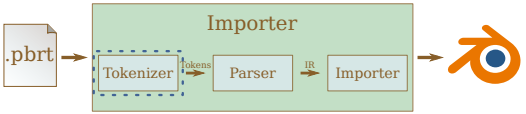
The Importer – `unit_import()`




```python
def unit_import(self, filename, import_settings):
    """Import a single PBRT file."""
    try:
        pbrt_settings, world = pbrt_parser(filename, import_settings)
        importer(pbrt_settings, world, import_settings)
        return {'FINISHED'}
    except Exception as e:
        traceback.print_exc()
        print(f"Unexpected exception caught: {e}")
        self.report({'ERROR'}, str(e.args[0]))
        return {'CANCELLED'}
```

Tokenizer

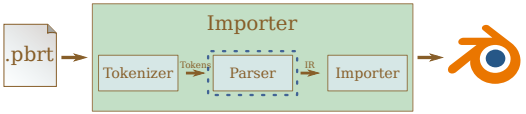
The Importer




```
import shlex

class Tokenizer(shlex.shlex):
    def __init__(self, file, name):
        super().__init__(instream=file,
                         infile=name,
                         punctuation_chars="[]")
        self.wordchars += "+"
```

Parser

The Importer




```
def parse_setting(token):
    """Parse a token in the PBRT settings context."""
    if token in ("Include", "Import"):
        parse_include()
    # ...
    else:
        raise ParseError(f"Unexpected token: '{token}'")
```

~600 lines of Python

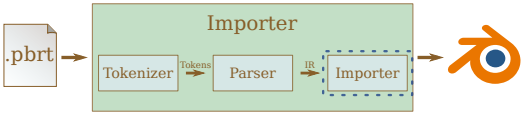
Intermediate Representation

The Importer


```
settings = { "animation": {} }
world = {
    "area_lights": [],
    "lights": [],
    "shapes": [],
    "instances": [],
    "materials": [],
    "textures": {},
    "media": {},
    "named_materials": {},
    "named_objects": {},
    "coordinate_systems": {},
}
```

Importer

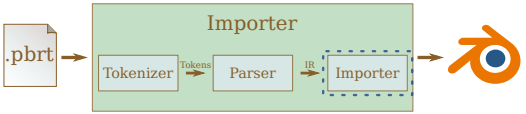
The Importer




```
def create(settings, world):
    """Import and convert all PBRT objects."""

    render_properties(settings)
    camera(settings, world)

    for m in world["materials"]:
        material(m)
    for s in world["shapes"]:
        shape(s)
    # ...
```

Import Shape

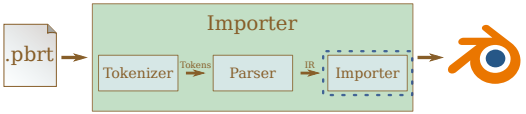
The Importer




```python
def shape(s):
    """Convert a PBRT shape to a Blender shape."""
    SHAPE_BUILDER = {
        "sphere": sphere,
        "trianglemesh": trianglemesh,
    }
    if s.type not in SHAPE_BUILDER:
        print(f"WARN: Unsupported shape: {s}")
        return
    build = SHAPE_BUILDER[s.type]
    build(s)
```

Import Sphere

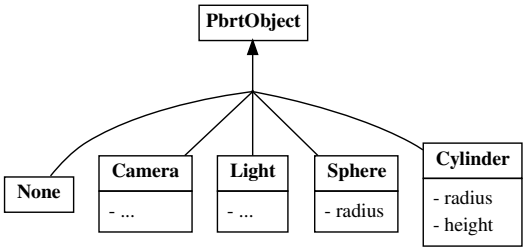
The Importer




```
def sphere(s):
    """Convert a PBRT sphere to a Blender shape."""
    r = s.parameters.get("radius", [1.0])[0]
    trf = mathutils.Matrix(s.start_transform)
    T, R, S = trf.decompose()
    opts = dict(radius=r,
                location=T,
                rotation=R.to_euler(),
                scale=S)
    bpy.ops.mesh.primitive_uv_sphere_add(**opts)
```

Sum Types and Object Orientation

```
enum PbrtObject
{
    None,
    Camera(...),
    Light(...),
    Sphere{radius: f32},
    Cylinder{radius: f32,
             height: f32},
}
```



Faking it in Blender – 1

```
class PbrtSphere(bpy.types.PropertyGroup):
    """PBRT Sphere properties."""

    radius: bpy.props.FloatProperty(
        name="radius",
        description="The sphere's radius.",
        default=1.0,
    )
```

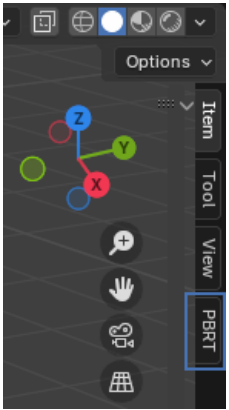
Faking it in Blender – 2

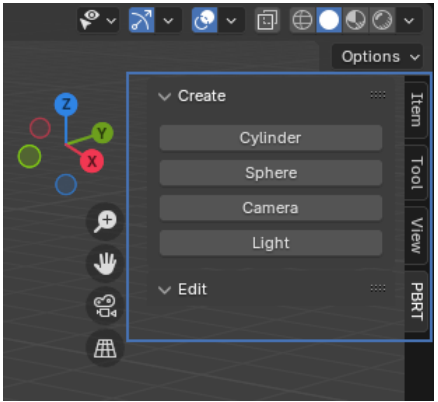
```
class PbrtShape(PropertyGroup):
    type: EnumProperty(
        name="type",
        items=[("none", "None", ""),
               ("camera", "Camera", ""),
               ("light", "Light", ""),
               ("sphere", "Sphere", ""),
               ("cylinder", "Cylinder", ""), ...],
        default="none")
    none: BoolProperty(name="none", default=False)
    camera: PointerProperty(type=PbrtCamera)
    light: PointerProperty(type=PbrtLight)
    sphere: PointerProperty(type=PbrtSphere)
    cylinder: PointerProperty(type=PbrtCylinder)
```

Selecting the Derived Property

```
def draw(self, ctx):
    """Draw the edit category buttons."""
    obj = ctx.selected_objects
    otype = getattr(obj.pbrt, obj.pbrt.type)
    for fname in otype.__annotations__.keys():
        if hasattr(otype, fname):
            self.layout.prop(otype, fname)
```

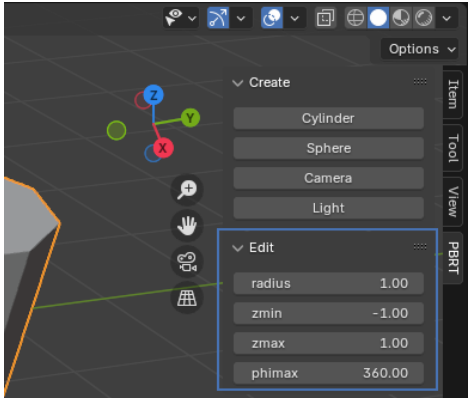
Proxy Object Panel

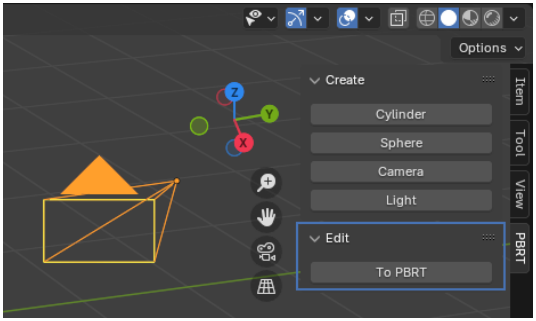




Updating and Converting Objects

Proxy Objects



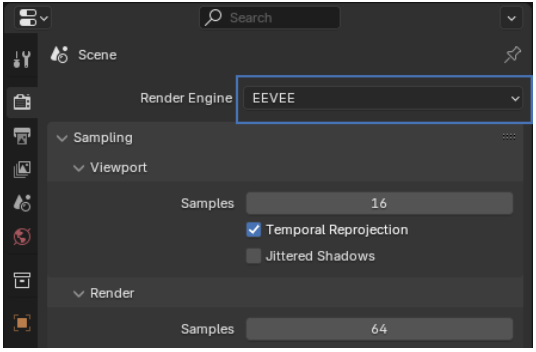


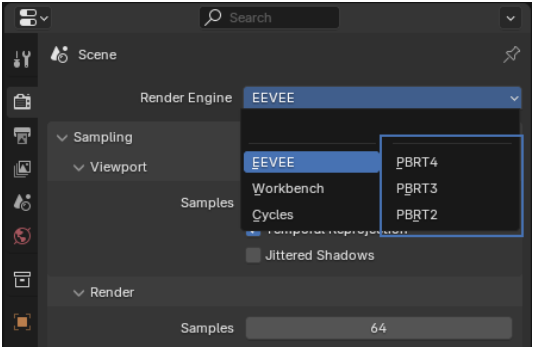
Render Properties

```
class RenderPBRT(bpy.types.RenderEngine):
    bl_idname = ""
    bl_label = ""

    # ...
```

```
class RenderPBRT4(RenderPBRT):
    bl_idname = "PBRT4"
    bl_label = "PBRT4"

class RenderPBRT3(RenderPBRT):
    bl_idname = "PBRT3"
    bl_label = "PBRT3"

class RenderPBRT2(RenderPBRT):
    bl_idname = "PBRT2"
    bl_label = "PBRT2"
```

Render Properties

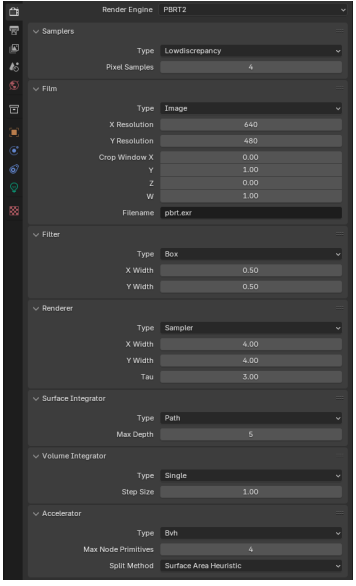




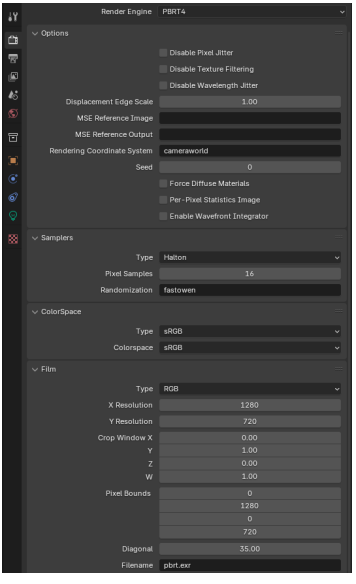
Render Properties

Panel Example

```
class PbrtV4Accelerator(PbrtProperty):
    """PBRT Accelerator properties."""

    type: bpy.props.EnumProperty(
        name="Accelerator Type",
        description="Accelerator Type",
        items=[("bvh", "BVH", ""),
               ("kdtree", "Kd-Tree", "")],
        default="bvh",
    )

    bvh: bpy.props.PointerProperty(type=PbrtV4BvhAccelerator)
    kdtree: bpy.props.PointerProperty(type=PbrtV4KdTreeAccelerator)
```


Render Engine PBRT2
Samplers
Type Lowdiscrepancy
Pixel Samples 4
Film
Type Image
X Resolution 640
Y Resolution 480
Crop Window X 0.00
Y 1.00
Z 0.00
W 1.00
Filename pbrt.exr
Filter
Type Box
X Width 0.50
Y Width 0.50
Renderer
Type Sampler
X Width 4.00
Y Width 4.00
Tau 3.00
Surface Integrator
Type Path
Max Depth 5
Volume Integrator
Type Single
Step Size 1.00
Accelerator
Type Bvh
Max Node Primitives 4
Split Method Surface Area Heuristic


Render Engine PBRT3
Samplers
Type Halton
Pixel Samples 16
Film
Type Image
X Resolution 640
Y Resolution 480
Crop Window X 0.00
Y 1.00
Z 0.00
W 1.00
Scale 1.00
Max Sample Luminance inf
Diagonal 35.00
Filename pbrt.exr
Filter
Type Box
X Width 0.50
Y Width 0.50
Integrator
Type Path
Max Depth 5
Pixel Bounds X 0
Y 640
Z 0
W 480
Russian Roulette threshold 1.00
Light Sampling Strategy spatial
Accelerator
Type BVH
Max Primitives per Node 4
Split Method Surface Area Heuristic


Render Engine PBRT4
Options
Disable Pixel Jitter
Disable Texture Filtering
Disable Wavelength Jitter
Displacement Edge Scale 1.00
MSE Reference Image
MSE Reference Output
Rendering Coordinate System cameraworld
Seed 0
Force Diffuse Materials
Per-Pixel Statistics Image
Enable Wavefront Integrator
Samplers
Type Halton
Pixel Samples 16
Randomization fastowen
ColorSpace
Type sRGB
Colorspace sRGB
Film
Type RGB
X Resolution 1280
Y Resolution 720
Crop Window X 0.00
Y 1.00
Z 0.00
W 1.00
Pixel Bounds 0
1280
0
720
Diagonal 35.00
Filename pbrt.exr

The Exporter

Earlier Work

The Exporter

- `https://github.com/giuliojiang/pbrt-v3-blender-exporter` (2.79)
- `https://github.com/stig-atle/io_scene_pbrt` (2.8x)
- `https://github.com/NicNel/bpbrt4` (2.9, Windows)

Back to the Exporter

What do we need?

- An Export *Operator*[1]
- An Export Panel[2] (optional)
- An Export Menu Entry[3] (optional)

[1] https://docs.blender.org/manual/en/latest/interface/operators.html
[2] https://docs.blender.org/api/current/bpy.types.Panel.html
[3] https://docs.blender.org/api/current/bpy.types.Menu.html

The Operator

The Exporter


```
class ExportPBRT(bpy.types.Operator, bpy_extras.io_utils.ExportHelper):
    """Blender Exporter class for PBRT scenes."""
    # ... Config variables...

    def draw(self, context):
        """Specify how to draw the Blender panel UI."""

    def execute(self, context):
        """Execute the PBRT export process."""
```

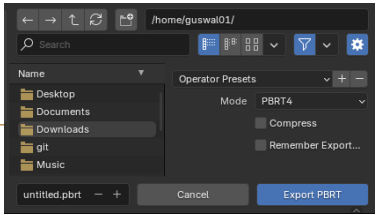
The Operator

The Exporter – Configuration Variables


```
class ExportPBRT(bpy.types.Operator, bpy_extras.io_utils.ExportHelper):
    """Blender Exporter class for PBRT scenes."""
    bl_idname = "export_scene.pbrt"
    bl_label = "Export PBRT"
    bl_options = {'PRESET'}
    filename_ext = ".pbrt"
    filter_glob: StringProperty(default="*.pbrt;*.pbrt.gz")
    mode: EnumProperty()
    compress: BoolProperty()
```

The Panel

The Exporter




```
def draw(self, context):
    """Specify how to draw the Blender panel UI."""
    sfile = context.space_data
    operator = sfile.active_operator
    layout = self.layout
    layout.use_property_split = True
    layout.use_property_decorate = False
    layout.prop(operator, "mode")
    layout.prop(operator, "compress")
```

The Operator

The Exporter – `execute()`


```
def execute(self, context):
    """Execute the PBRT export process."""
    with open(self.filepath, "w", encoding="utf-8") as f:
        with contextlib.redirect_stdout():
            export_camera(self.mode, context)
            export_renderer(self.mode, context)
            export_world(self.mode, context)
    return {"FINISHED"}
```

The Operator

The Exporter – `export_renderer()`


```
def export_renderer(mode, ctx):
    """Export PBRT renderer settings."""
    prop = getattr(ctx.scene.PbrtRenderProperties, mode)
    prop.to_pbrt()
```

The Operator

The Exporter – to_pbrt()


```
def to_pbrt(self):
    """Print a PBRT representation of this object."""
    ptype = getattr(self, self.type)
    print(f"{self.NAME} \"{self.type}\"")
    for name in ptype.__annotations__.keys():
        prop = ptype.rna_type.properties[name]
        val = getattr(ptype, name)
        ntype = pbrt_type(val, prop)
        val = pbrt_value(val, prop)
        print(f"\"{ntype} {name}\" [{val}]")
```

The Operator

The Exporter – `export_world()`


```
def export_world(engine, ctx):
    """Export the PBRT world objects."""
    print("WorldBegin")
    for obj in bpy.data.objects:
        if obj.type == "LIGHT":
            export_light(obj, engine, ctx)
        else:
            export_shape(obj, engine, ctx)
    if engine in ("PBRT2", "PBRT3"):
        print("WorldEnd")
```

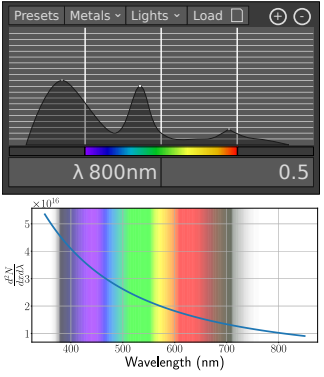
The Operator

The Exporter – `export_shape()`


```
def export_shape(obj, mode, ctx):
    """Export a PBRT shape."""
    print("AttributeBegin")
    print("    Transform %s" % (get_transform(obj)))
    print("    Material ..." % (get_material(obj)))
    if obj.pbrt.type != "none":
        export_pbrt_object(obj, mode, ctx)
    elif obj.type == "MESH":
        # ...
    print("AttributeEnd")
```

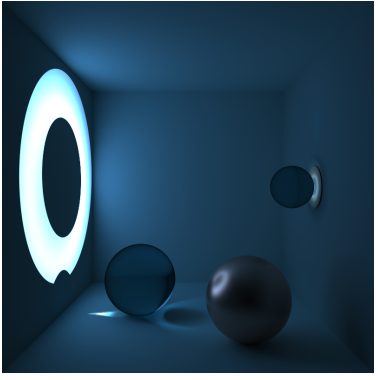
Missing Feature(s)

- Spectral Properties



Conclusion

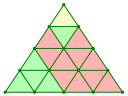
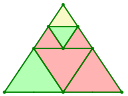
1. PBRT
2. Importers / Exporters
3. Render Properties and Proxies



Thanks for Listening!

Questions and Answers

- Thanks for listening!
- Questions and Answers

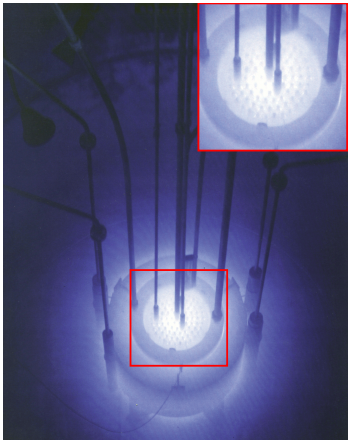



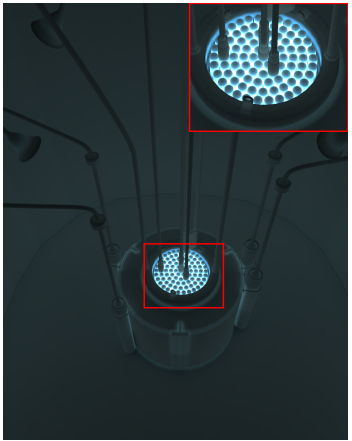

Making Blender ♡ PBRT
Create your own Importers and Exporters

- https://gustafwaldemarson.com/
  - https://gustafwaldemarson.com/pages/publications/
- gustaf.waldemarson@cs.lth.se
- gustaf.waldemarson@arm.com

The End

Superluminal Reactor Rendering

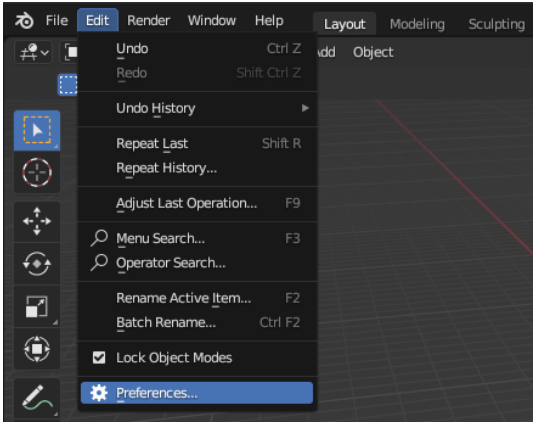




Testing Pipeline

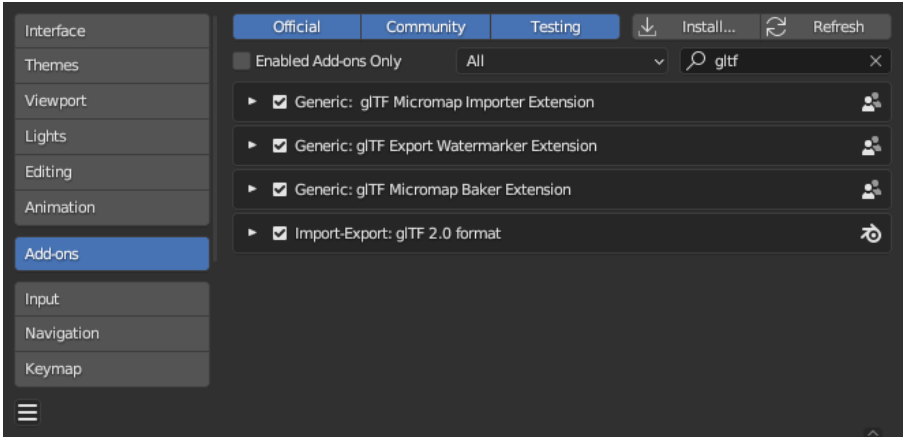
Intermission: Installing and Enabling Addons

Blender <= 4.1



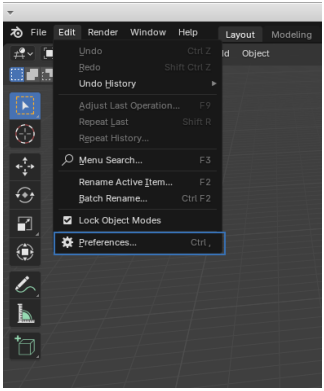
Intermission: Installing and Enabling Addons

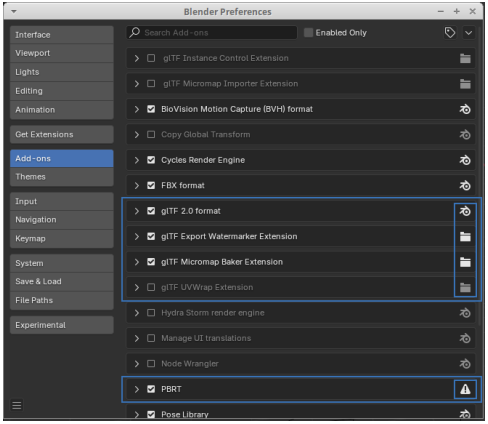
Blender <= 4.1



Intermission: Installing and Enabling Addons

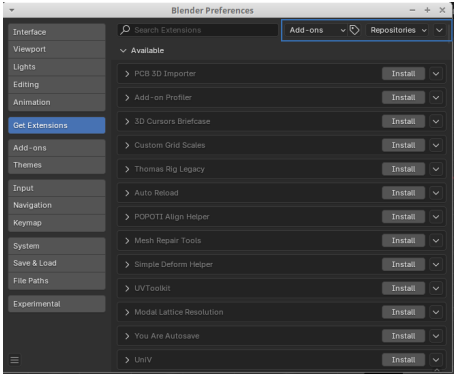
Blender 4.2+ (Legacy Addons)

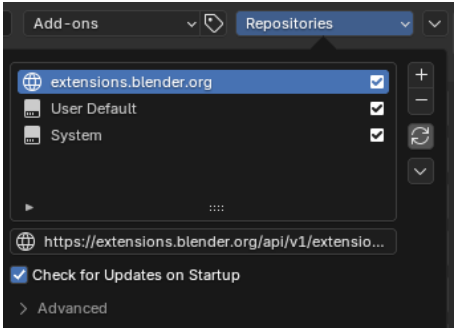




Intermission: Installing and Enabling Addons

Blender 4.2+ (Extensions)





References I

[1] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. 4th. MIT Press, 2023.

[2] Gustaf Waldemarson. *Handling Custom Data in glTF Files with Exporter/Importer Plugins*. The Blender Foundation, Youtube. 2023. URL: https://youtu.be/4fBGM8qc21M?t=1783.

[3] Gustaf Waldemarson and Michael Doggett. "Photon Mapping Superluminal Particles". In: *Eurographics 2020 - Short Papers*. Ed. by Alexander Wilkie and Francesco Banterle. The Eurographics Association, 2020. ISBN: 978-3-03868-101-4. DOI: 10.2312/egs.20201004.

[4] Gustaf Waldemarson and Michael Doggett. "Succinct Opacity Micromaps". In: *Proc. ACM Comput. Graph. Interact. Tech.* 7.3 (Aug. 2024). DOI: 10.1145/3675385. URL: https://doi.org/10.1145/3675385.