

The Intel logo is displayed in white text on a blue square background in the top left corner of the slide.

Intel[®] Inspector

Addition: API usage

CERN, March 3rd 2022

Heinrich Bockhorst, Intel

```
class MirrorX(bpy.types.Operator):
    """This adds an X mirror to the selected object"""
    bl_idname = "object.mirror_mirror_x"
    bl_label = "Mirror X"

    @classmethod
    def poll(cls, context):
        return context.active_object is not None

class MirrorX(bpy.types.Operator):
    """This adds an X mirror to the selected object"""
    bl_idname = "object.mirror_mirror_x"
    bl_label = "Mirror X"

    @classmethod
    def poll(cls, context):
        return context.active_object is not None
```

```
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

#selection at the end -add back the deselected mirror ob
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected = %s" % modifier_ob)
except:
    print("please select exactly two objects, the last one p")

class MirrorX(bpy.types.Operator):
    """This adds an X mirror to the selected object"""
    bl_idname = "object.mirror_mirror_x"
    bl_label = "Mirror X"

    @classmethod
    def poll(cls, context):
        return context.active_object is not None
```

Custom memory allocation

```
#include <ittnotify.h>

__itt_heap_function my_allocator;
__itt_heap_function my_reallocator;
__itt_heap_function my_freer;

void* my_malloc(size_t s)
{
    void* p;

    __itt_heap_allocate_begin (my_allocator, s, 0);
    p = user_defined_malloc (s);
    __itt_heap_allocate_end (my_allocator, &p, s, 0);

    return p;
}
... // Do similar markup for custom "realloc" and "free" operations

// Call this init routine before any calls to user defined allocators
void init_itt_calls()
{
    my_allocator = __itt_heap_function_create("my_malloc", "mydomain");
    my_reallocator = __itt_heap_function_create("my_realloc", "mydomain");
    my_freer = __itt_heap_function_create("my_free", "mydomain");
}
```

Collection control APIs

API	Description
<code>void __itt_suppress_push(unsigned int etype)</code>	Stop analyzing for errors on the current thread
<code>void __itt_suppress_pop(void)</code>	Resume analysis
<code>void __itt_suppress_mark_range(__itt_suppress_mode_t mode, unsigned int etype, void * address, size_t size);</code>	Suppress or unsuppress error detection for the specific memory range (object).
<code>void __itt_suppress_clear_range(__itt_suppress_mode_t mode, unsigned int etype, void * address, size_t size);</code>	Clear the marked memory range

User-Defined Synchronization APIs

API	Description
<code>void __itt_sync_acquired(void *addr)</code>	Notify Intel Inspector that synchronization object is acquired by current thread
<code>void __itt_sync_releasing(void *addr)</code>	Notify that the code is about to release the specified synchronization object
<code>void __itt_sync_destroy(void *addr)</code>	Tell the Intel Inspector that the synchronization object will not be used again, so the Intel Inspector can dispose of bookkeeping information associated with this object.

Documentation

- API calls (user guide)

<https://www.intel.com/content/www/us/en/develop/documentation/inspector-user-guide-linux/top/api-support.html>