Location Tracker

```
import xbmc, xbmcaddon, xbmcgui, os, re, threading, pyxbmct

ADDON = xbmcaddon.Addon()
addonpath = ADDON.getAddonInfo("path")
if (__name__ == "__main__"):
print("filter addon starting")



def from_hms(timeString):
timeString = timeString.replace(",", ".") # in .srt format decimal seconds use a comma
time = timeString.split(":")
if len(time) == 3: # has hours
return (int(time[0]) * 3600) + (int(time[1]) * 60) + float(time[2])
elif len(time) == 2: # minutes and seconds
return (int(time[0]) * 60) + float(time[1])
else: # only seconds
return float(time[0])

def parse_tag_action_info(rawText): # 2 tags at the same time?
category = ""
```

```
severity = ""
action = ""

textWithoutComment = rawText.split(" #")[0] # Eliminate tag comments
textArray = textWithoutComment.split("=")

category = textArray[0]

if textArray[1] == "high":
severity = 3
elif textArray[1] == "medium":
severity = 2
elif textArray[1] == "low":
severity = 1

if len(textArray) == 2 or textArray[2] == "both":
action = "skip"
elif textArray[2] == "video":
action = "blank"
elif textArray[2] == "audio":
action = "mute"

return [category, severity, action]
```

```
def parse_filter_file_text(fileText):

# Modified from code by nqngo at Stack Overflow, used to separate timestamps in the filter file from the tag descriptions

# https://stackoverflow.com/questions/23620423/parsing-a-srt-file-with-regex

result = re.findall("(\d+:\d+:\d+.*\d* --> \d+:\d+:\d+.*\d*)\s+(.+)", fileText)


allCuts = []

for myTuple in result:

currentCut = {}

times = myTuple[0].split(" --> ")

currentCut["startTime"] = from_hms(times[0])

currentCut["endTime"] = from_hms(times[1])


actionInfo = parse_tag_action_info(myTuple[1])

currentCut["category"] = actionInfo[0]

currentCut["severity"] = actionInfo[1]

currentCut["action"] = actionInfo[2]

allCuts.append(currentCut)


return allCuts
```

```
userSettings = {}


def update_user_settings():

categoryIdList = ["commercial", "advertBreak", "consumerism", "productPlacement", "discrimination", "ableism", "adultism", "antisemitism", "genderism", "homophobia", "misandry", "misogyny", "racism", "sexism", "sizeism", "supremacism", "transphobia", "xenophobia", "dispensable", "idiocy", "tedious", "drugs", "alcohol", "antipsychotics", "cigarettes", "depressants", "gambling", "hallucinogens", "stimulants", "fear", "accident", "acrophobia", "aliens", "arachnophobia", "astraphobia", "aviophobia", "chemophobia", "claustrophobia", "coulrophobia", "cynophobia", "death", "dentophobia", "emetophobia", "enochlophobia", "explosion", "fire", "gerascophobia", "ghosts", "graves", "hemophobia", "hylophobia", "melissophobia", "misophonia", "musophobia", "mysophobia", "nosocomephobia", "nyctophobia", "siderodromophobia", "thalassophobia", "vampires", "language", "blasphemy", "nameCalling", "sexualDialogue", "swearing", "vulgarity", "nudity", "bareButtocks", "exposedGenitalia", "fullNudity", "toplessness", "sex", "adultery", "analSex", "coitus", "kissing", "masturbation", "objectification", "oralSex", "premaritalSex", "promiscuity", "prostitution", "violence", "choking", "crueltyToAnimals", "culturalViolence", "desecration", "emotionalViolence", "kicking", "massacre", "murder", "punching", "rape", "slapping", "slavery", "stabbing", "torture", "warfare", "weapons"]

global userSettings

userSettings = {}

for category in categoryIdList:

userSettings[category] = ADDON.getSetting(category)


# Modified from isSkipped function from "content2.js" from VideoSkip

def is_tag_active(tag, userSettings):
```

```
category = tag["category"]

return int(tag["severity"]) + int(userSettings[category]) > 3


activeCuts = []


# Modified from isSkipped function from "content2.js" from VideoSkip

def apply_filters(allCuts, userSettings):

global activeCuts

activeCuts = []

for cut in allCuts:

if is_tag_active(cut, userSettings):

activeCuts.append(cut)


def load_filter_file():

try:

filePath = xbmc.Player().getPlayingFile().rsplit(".", 1)[0] + ".mcf"

fileInput = open(filePath, "r")

fileText = fileInput.read()

allCuts = parse_filter_file_text(fileText)

update_user_settings()

global userSettings

apply_filters(allCuts, userSettings)

except OSError:
```

```
print("Movie Content Filter Add-on: Unable to find or open MCF file")

global activeCuts

activeCuts = [] # To prevent the current video from using the cuts from the
previous video that had cuts


# Modified from the LazyMonitor class from "service.py" from LazyTV

class AppMonitor(xbmc.Monitor):

def __init__(self, *args, **kwargs):

xbmc.Monitor.__init__(self)


def onSettingsChanged(self):

update_user_settings()


monitor = AppMonitor()


class FamilyMovieActNotice(object):

def __init__(self):

self.showing = False

self.window = xbmcgui.Window(12005)


# Arbitrary values for position and size for now

origin_x = 300

origin_y = 400
```

```
window_w = int(xbmc.getInfoLabel("System.ScreenWidth")) * 2

window_h = int(xbmc.getInfoLabel("System.ScreenHeight")) * 4


#main window

self._disclaimer = xbmcgui.ControlTextBox(origin_x, origin_y, window_w, window_h) # How to add shadow color?

self._disclaimer.setText(ADDON.getLocalizedString(32106))


def show(self):

if not self.showing:

self.showing=True

self.window.addControl(self._disclaimer)


def hide(self):

if self.showing:

self.showing=False

self.window.removeControl(self._disclaimer)


def _close(self):

if self.showing:

self.hide()

self.window.clearProperties()
```

```
def display_legal_notice():

if "legalNotice" not in locals():

legalNotice = FamilyMovieActNotice()

legalNotice.show()


def remove_notice(legalNotice):

legalNotice.hide()

legalNotice._close()


timer = threading.Timer(6.0, lambda: remove_notice(legalNotice))

timer.start()


def check_for_editor():

if(ADDON.getSetting("editorActive") == "true"):

# Add editor window

pass


class XBMCPlayer(xbmc.Player):

def onAVChange(self):

load_filter_file()

display_legal_notice()

check_for_editor()
```

```
player = XBMCPlayer()


class OverlayBlankScreen(object):
def __init__(self):
self.showing = False
self.window = xbmcgui.Window(12005) # Inheriting from 12005 keeps the black background from overlaying the interface
origin_x = 0
origin_y = 0
# Since Kodi seems to usually report a smaller screen width and height than there really is, multiplying the values can be a hacky way to make sure the whole screen is covered when hiding the video
window_w = int(xbmc.getInfoLabel("System.ScreenWidth")) * 100
window_h = int(xbmc.getInfoLabel("System.ScreenHeight")) * 100


#main window
self._background = xbmcgui.ControlImage(origin_x, origin_y, window_w, window_h, os.path.join(addonpath,"resources","skins","default","media","black-background.jpg"))


def show(self):
if not self.showing:
self.showing=True
self.window.addControl(self._background)
```

```
def hide(self):

if self.showing:

self.showing=False

self.window.removeControl(self._background)


def _close(self):

if self.showing:

self.hide()

self.window.clearProperties()

#print("OverlayBlankScreen window closed")


prevAction = ""


# Execute filters during playback, derived and modified from anonymous function
in "content1.js" from VideoSkip (version 0.4.1), originally "content2.js"

def do_the_filtering(prevAction, activeCuts, blankScreen):

startTime = 0

endTime = 0

action = ""

tempAction = ""

for tag in activeCuts:

startTime = tag["startTime"]
```

```
endTime = tag["endTime"]
currentTime = xbmc.Player().getTime()
if currentTime > startTime and currentTime < endTime:
tempAction = tag["action"]
else:
tempAction = ""

if tempAction == "skip": # Retain the strongest action valid for the current time. Hierarchy: skip > blank > mute
action = "skip"
break # Can't get any stronger, so stop looking for this time
elif tempAction == "blank":
if action != "skip":
action = tempAction
elif tempAction == "mute":
if action == "blank":
action = "skip"
else:
action = "mute"

if action == prevAction:
return prevAction
elif action == "skip":
```

```
xbmc.Player().seekTime(float(endTime) + 0.1)

elif action == "blank":

blankScreen.show()

elif action == "mute":

xbmc.executebuiltin("SetVolume(0)")

else:

xbmc.executebuiltin("SetVolume(100)")

blankScreen.hide()

prevAction = action

return prevAction


while not monitor.abortRequested():

if monitor.waitForAbort(0.01):

break

if xbmc.getCondVisibility("Player.HasMedia"):

if "blankScreen" not in locals():

blankScreen = OverlayBlankScreen()

prevAction = do_the_filtering(prevAction, activeCuts, blankScreen)


blankScreen._close()



# To Do List:
```

# Family Movie Act of 2005 notice (including black background behind text for readability and keeping the right Z-index, if needed, plus the right position and size)

# Filtering editor (activate/deactivate through add-on settings)