Implementation Plan - Animation Modernization (Phase 2.5)

Version: 1.0.0 **Target:** 0.94.2-Beta

1. Context & Rationale

We are executing Phase 2.5 of the Roadmap. The goal is to eliminate the project's dependency on `ffmpeg` and `matplotlib` for animations by standardizing on the new Plotly-based `plot_interactive_animation.py`.

This plan performs two critical actions:

1. **Cleanup:** Removes the legacy `plot_hourly_animation.py`, which generates MP4 videos and relies on `imageio` and `matplotlib`.
2. **Verification Update:** Updates `run_regression_test.py` to support the new `.html` output format. The test script must treat `.html` files as text to allow for `diff` comparison, ensuring the Plotly markup remains stable across runs. It also removes obsolete logic related to checking sequential animation frames (`_frame_001.png`), which are no longer generated.

2. Affected Files

- `run_regression_test.py`: Update to handle `.html` comparison and remove legacy frame logic.
- `contest_tools/reports/plot_hourly_animation.py`: **DELETE**.

3. Surgical Changes

File: `run_regression_test.py`

Goal: Enable text-based comparison for `.html` files and remove obsolete logic for sequential frame checking.

1. **Update `_parse_sequence_filename`**:
   - Remove the logic that looks for `_frame_(\d{3})` patterns. This is specific to the old MP4 generation process.
2. **Update `compare_outputs`**:
   - Add `.html` to the list of file extensions processed in the "Regular Files" block (alongside `.txt`, `.json`, etc.).
   - Add a specific `elif` block for `.html` files to read them as text (utf-8) and perform a `difflib.unified_diff`.

File: `contest_tools/reports/plot_hourly_animation.py`

Goal: Remove legacy code.

- **Action:** Delete file.

---

4. Surgical Change Verification (`diff`)

`run_regression_test.py`

___CODE_BLOCK___python --- BEGIN DIFF --- @@ -23,17 +23,12 @@ def _parse_sequence_filename(filename: str): """ Checks if a filename is part of a sequence and extracts its number. Returns a tuple of (group_name, number) or (None, None). """

- frame_match = re.search(r'*frame*(\d{3}).(txt|json)$', filename, re.IGNORECASE)
- if frame_match:
- `    return "Animation Frames", int(frame_match.group(1))`
- page_match = re.search(r'*Page*(\d+)*of*(\d+).(png|txt|json)$', filename, re.IGNORECASE) if page_match: return "Paginated Reports", int(page_match.group(1))

  return None, None

def _parse_adif_for_comparison(filepath: str) -> List[Dict[str, str]]: @@ -142,13 +137,13 @@ relative_path = os.path.relpath(item_path, reports_dir) baseline_path = os.path.join(archive_dir, relative_path) failures = []

```
    if not os.path.exists(baseline_path):
```

- `        if item_path.endswith(('.png', '.mp4', '.txt', '.adi', '_processed.csv')):`
- `        if item_path.endswith(('.png', '.html', '.txt', '.adi', '_processed.csv')):`

```
            failures.append(f"Missing Baseline: {relative_path}")
        if failures:
            all_failures.extend(failures)
        continue

    elif item_path.endswith('.json'):
        try:
```

@@ -198,12 +193,27 @@ except Exception as e: failures.append(f"File: {relative_path}\nERROR during CSV diff: {e}")

```
    elif item_path.endswith('.adi'):
        try:
            qsos_new = _parse_adif_for_comparison(item_path)
            qsos_base = _parse_adif_for_comparison(baseline_path)

            if qsos_new != qsos_base:
                # --- New Detailed Comparison Logic ---
                summary_lines = [f"File: {relative_path}"]
```

@@ -260,13 +270,29 @@ except Exception as e: failures.append(f"File: {relative_path}\nERROR during ADIF diff: {e}")

- `elif item_path.endswith('.txt'):`
- `elif item_path.endswith(('.txt', '.html')):`

```
            try:
                with open(item_path, 'r', encoding='utf-8') as f_new:
                    lines_new = f_new.readlines()
                with open(baseline_path, 'r', encoding='utf-8') as f_base:
                    lines_base = f_base.readlines()

                if ignore_whitespace:
                    lines_new = [" ".join(line.split()) + '\n' for line in lines_new]
                    lines_base = [" ".join(line.split()) + '\n' for line in lines_base]

                diff = list(difflib.unified_diff(
                    lines_base, lines_new,
                    fromfile=f'a/{relative_path}', tofile=f'b/{relative_path}'
                ))

                if diff:
                    failures.append(f"File: {relative_path}\n" + "".join(diff))
            except Exception as e:
                failures.append(f"File: {relative_path}\nERROR during diff: {e}")

        if failures:
            all_failures.extend(failures)
```

--- 3. Process sequence files (new summary behavior) ---

--- END DIFF --- **CODE_BLOCK**

---

5. Affected Modules Checklist

- `run_regression_test.py`: Primary target.
- `contest_tools/reports/plot_hourly_animation.py`: Being deleted.
- **Systemic Check:** No other test runners or report generators utilize the legacy frame logic or require `.html` comparison logic at this time.

6. Pre-Flight Check

- **Inputs:** `run_regression_test.py` (Version 0.90.1-Beta), `plot_hourly_animation.py` (Version 0.90.4-Beta).
- **Expected Outcome:**
  - `run_regression_test.py` will correctly diff `.html` files produced by Plotly.
  - `run_regression_test.py` will no longer scan for `_frame_NNN` files.
  - `plot_hourly_animation.py` will be removed from the project.
- **Mental Walkthrough:** I have verified that removing the frame regex will not affect paginated reports (handled by a separate regex block). I have confirmed that adding `.html` to the text processing block uses standard `difflib`, which is appropriate for markup.
- **State Confirmation:** The `Confirmed` prompt will be used.
- **Impact Analysis:** Deleting `plot_hourly_animation.py` will break any scripts explicitly calling `--report hourly_animation` or relying on MP4 output. This is the intended deprecation path.
- **Protocol Compliance:** This plan adheres to the Surgical Modification protocol (Protocol 2.5) and the File Purge Protocol (Protocol 3.6).